

**DEVELOPMENT OF INSTRUMENTED BED  
MONITORING SYSTEM**

**MERCEDES CHAN SIMH PEH**

**UNIVERSITI TUNKU ABDUL RAHMAN**

**DEVELOPMENT OF INSTRUMENTED BED  
MONITORING SYSTEM**

**MERCEDES CHAN SIMH PEH**

**A project report submitted in partial fulfilment of the  
requirements for the award of Bachelor of Engineering  
(Honours) Biomedical Engineering**

**Lee Kong Chian Faculty of Engineering and Science  
Universiti Tunku Abdul Rahman**

**September 2023**

**DECLARATION**

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature :  \_\_\_\_\_

Name : Mercedes Chan Simh Peh \_\_\_\_\_

ID No. : 1906112 \_\_\_\_\_

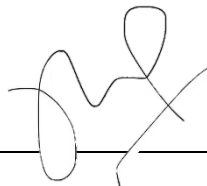
Date : 15/09/2023 \_\_\_\_\_

**APPROVAL FOR SUBMISSION**

I certify that this project report entitled **“DEVELOPMENT OF INSTRUMENTED BED MONITORING SYSTEM”** was prepared by **MERCEDES CHAN SIMH PEH** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Biomedical Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature :



Supervisor :

Mr. Chong Yu Zheng

Date :

18/09/2023

Signature :



Co-Supervisor :

Dr. Chan Siow Cheng

Date :

18/09/2023

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2023, MERCEDES CHAN SIMH PEH. All right reserved.

## **ACKNOWLEDGEMENTS**

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Mr Chong for his invaluable advice, guidance, and his enormous patience throughout the development of the device.

In addition, I would also like to express my gratitude to my loving parents and friends who had helped and given me encouragement throughout the course of the last six months.

## ABSTRACT

Numerous distressing Hospital-Acquired Conditions (HACs), such as falls, and pressure injuries, predominantly occur within the vicinity of the hospital bed. Despite much effort, these conditions are still prevalent today. Current prevention practices for these HACs involve a combination of staff protocol and technological assistive devices. However, these measures have a heavy reliance on the medical staff, with vulnerabilities exacerbated by staff shortages, especially in overcrowded medical facilities. Henceforth, this project aims to develop an instrumented bed monitoring system that can function as a bed-exit alarm whilst mitigating pressure injuries via the continuous monitoring of movement. The system is comprised of a central processing unit, touchscreen user interface and a multilayered sensor array, in the form of a mattress protector sheet. The sensor sheet is a qualitative pressure array consisting of 128 points, arranged in an eight by sixteen matrix and features two layers: one with eight vertical sensing lines and the other with sixteen horizontals. These sensor lines are thin stainless-steel wires sewn into cloth with the help of a sewing machine and have proven to be incredibly sensitive to touch. Scanning of the array is done in real time, allowing for continuous monitoring of the patient's status. Moreover, the patient status can be viewed on the colour touchscreen at the bedside terminal or remotely through a web browser on a computer or handphone. Ultimately, the nurses will be notified when anomalies are identified, and the alarms are triggered. These are instances such as the absence of bodily movement for more than 2 hours or when the patient moves to make an unauthorised exit off the bed, enabling timely interventions, where in this case it is to either reposition the patient to prevent bed sores or aid the frail patients from falling. The current project prototype can effectively detect items that are heavier than 350g in weight and is 70.3% accurate in reproducing the sleeping profile of the user on the instrumented bed. Although more work is required to improve the bed array sensing accuracy, the system has no issues performing its intended functions in mitigating preventable suffering caused by HACs, as proven by the high efficacy rates of the alarms and alerts.

## TABLE OF CONTENTS

<b>DECLARATION</b>	<b>i</b>
<b>APPROVAL FOR SUBMISSION</b>	<b>ii</b>
<b>ACKNOWLEDGEMENTS</b>	<b>iv</b>
<b>ABSTRACT</b>	<b>v</b>
<b>TABLE OF CONTENTS</b>	<b>vi</b>
<b>LIST OF TABLES</b>	<b>ix</b>
<b>LIST OF FIGURES</b>	<b>x</b>
<b>LIST OF APPENDICES</b>	<b>xiii</b>

### CHAPTER

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 General Introduction	1
	1.2 Importance of the Study	1
	1.3 Problem Statement	2
	1.4 Aim and Objectives	3
	1.5 Scope and Limitation of the Study	3
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>4</b>
	2.1 Introduction	4
	2.2 Prevalence of Bed-related Hospital Incidents	4
	2.2.1 Prevalence of Unauthorised Bed Exits	4
	2.2.2 Prevalence of Falls in a Hospital Setting	5
	2.2.3 Prevalence of Pressure Injuries	7
	2.3 Current Prevention Measures	9
	2.3.1 Current Prevention Practices for Combatting Pressure Injuries	9
	2.3.2 Bed-Exit & Fall Prevention Measures	10
	2.4 Sensor Designs	14
	2.5 Summary	15



<b>3</b>	<b>METHODOLOGY AND WORK PLAN</b>	<b>16</b>
3.1	Introduction	16
3.2	Selection of Main Components	16
3.2.1	ESP32	16
3.2.2	I/O Expanders	17
3.2.3	Touchscreen	18
3.3	Sensor Design	19
3.3.1	Commercial Grade Sensors	19
3.3.2	Design of the Sensor Array	21
3.4	Electronic Design & Schematic	25
3.4.1	Printed Circuit Boards	25
3.4.2	Device Design and Layout	26
3.4.3	Circuitry Design & Schematic	27
3.4.4	PCB Design	31
3.5	Casing Design	33
3.6	Software Development	34
3.6.1	Sensor Array Reporting Process	34
3.6.2	Analysing Data for Triggering Alarms	35
3.6.3	Touchscreen Communication	36
3.6.4	Touchscreen GUI	37
3.6.5	Web-based User Interface	39
3.7	Prototype Fabrication	40
<b>4</b>	<b>RESULTS &amp; DISCUSSION</b>	<b>42</b>
4.1	Introduction	42
4.2	Sensor Array Efficacy	42
4.2.1	Sensor Node Functionality Assessment	42
4.2.2	Minimum Weight Required for Detection	43
4.2.3	Sensor Array Working Current	44
4.2.4	Stray Points on Sensor Array	44
4.2.5	Sensor Array Reproduction Accuracy	45
4.3	Efficacy of system alerts	47
4.3.1	Time-to-turn-Patient Alarm	47
4.3.2	Bed-Exit Alerts	48
4.3.3	Rate of Scanning Bed Array	49

<b>5</b>	<b>CONCLUSIONS AND RECOMMENDATIONS</b>	<b>50</b>
5.1	Conclusions	50
5.2	Recommendations for Future Work	50
	<b>REFERENCES</b>	<b>52</b>
	<b>APPENDICES</b>	<b>59</b>

**LIST OF TABLES**

Table 2.3.1: Comparison of Pressure-Mats Bed-Exit Alarm Systems	13
Table 4.2.1: Sensor Node Assessment	42
Table 4.2.2: Successful Detections according to Weight	43
Table 4.2.3: Confusion Matrix for Average Human Laying-down	46

## LIST OF FIGURES

Figure 2.2.1:	Common Sites for Pressure Sores	7
Figure 2.2.2:	NPIAP Staging for Pressure Sores	8
Figure 2.3.1:	Pressure Relieving Air Mattresses: Static (Left), Alternating (Right)	10
Figure 2.3.2:	Stryker Chaperone Bed Exit System	12
Figure 3.2.1:	ESP-WROOM-32 Board Pinout Diagram	17
Figure 3.2.2:	I/O Expanders - MCP23008 (left) and MCP23017 (right)	17
Figure 3.2.3:	Nextion Intelligent NX8048P050-011C HMI Capacitive Display	18
Figure 3.3.1:	Examples of Thin-film Sensors and Flat Strain Gauge Sensors	20
Figure 3.3.2:	Bed Array Sensor Design on a Single Bed	21
Figure 3.3.3:	Close-up of Sensor Design	22
Figure 3.3.4:	Prototype of a Single Sensor Node with 0.2 mm Stainless Steel	23
Figure 3.3.5:	VT8 Prototype Sensor Layer: Eight vertical sensor lines.	24
Figure 3.3.6:	HZ16 Prototype Sensor Layer: Sixteen horizontal sensor lines	24
Figure 3.4.1:	Device Set-up on a Single-bed Configuration	26
Figure 3.4.2:	Schematic Connections to Main Microcontroller - ESP32	27
Figure 3.4.3:	Schematic Circuitry for 12V Power Socket, LED & 5V Regulator	28
Figure 3.4.4:	Schematic Connections to Peripheral Boards	28
Figure 3.4.5:	Schematic Connections to MCP23008 on VT8	29
Figure 3.4.6:	Schematic Connections to MCP23017 on HZ16	29
Figure 3.4.7:	Schematic Connections of VT8 to Vertical Sensor Array	30

Figure 3.4.8:	Schematic Connections of HZ16 to the Horizontal Sensor Array	30
Figure 3.4.9:	PCB for Main Board: Front View (Left), Back View (Right)	31
Figure 3.4.10:	3D-View of Main Board	31
Figure 3.4.11:	PCB for VT8 Board: Front View (Left), Back View (Right)	32
Figure 3.4.12:	PCB for HZ16 Board: Front View (Left), Back View (Right)	32
Figure 3.5.1:	3D-printed Front Case: Top View (Left), Bottom View (Right)	33
Figure 3.5.2:	3D-printed Back Plate	33
Figure 3.6.1:	Partial code for scan_bedArray()	34
Figure 3.6.2:	Partial Main Loop Code for Bed Movement	35
Figure 3.6.3:	Partial Main Loop Code for Bed Occupancy	36
Figure 3.6.4:	Partial Main Loop Code for Touchscreen Communication	36
Figure 3.6.5:	Code for handleTouchEvent()	37
Figure 3.6.6:	Touchscreen User Interface Main Page	37
Figure 3.6.7:	Touchscreen User Interface System Settings Page	38
Figure 3.6.8:	Touchscreen User Interface Network Settings Page	38
Figure 3.6.9:	Touchscreen User Interface Date & Time Page	39
Figure 3.6.10:	Web-based User Interface	39
Figure 3.7.1:	Device console attached to headboard	40
Figure 3.7.2:	Assembled sensor array with stacked layers	40
Figure 3.7.3:	Device user interface: Web-based (left), Touchscreen (Right)	41
Figure 4.2.1:	Number of Successful detections by Weight class	43
Figure 4.2.2:	Example of Sensor Array Reproduction	45
Figure 4.2.3:	Sensor Array Reproduction of Side-laying Position	46

Figure 4.3.1:	Pie Chart of Movement Detection for TTTP Alarm	47
Figure 4.3.2:	Pie Chart of 'Getting ON' for Bed-Exit Alert	48
Figure 4.3.3:	Pie Chart of 'Getting OFF' for Bed-Exit Alert	48

**LIST OF APPENDICES**

Appendix A:	Turnitin Report	59
Appendix B:	Microcontroller Code	60

## CHAPTER 1

### INTRODUCTION

#### 1.1 General Introduction

Hospitals wards are an integral part of any hospital and the healthcare system as a whole. They are designed to provide a safe and comfortable environment ideal for rest while undergoing treatment or recuperation. It is here where most patients will spend majority of their time while at the hospital. As such, they are considered the hotspots of the hospital with high amounts of traffic, from medical professional, patients, visitors and more. However, a significant concern, in the form of hospital-acquired conditions (HAC), has come to attention over the last few decades.

Hospital-acquired conditions (HACs) are patient complications that arise during the course of hospitalization that were previously not present at the time of admission. These conditions have led to more than 3000 patient deaths in 2016. According to a report published by IBM Watson Health, patients who contracted HACs were reported to have an increased mortality risk by 72 percent, along with an extended stay of 8 days on average. In 2016, more than 2 billion US dollars were incurred in operational costs due to HACs, roughly translating to about forty thousand USD per patient.

While there are many types of HACs, falls and pressure injuries are two prevalent conditions that are closely related to the hospital environment, ward and bed. They are considered high priority complications and have huge potential in severely affecting an individual's life. Hospitals around the world have been striving to prevent them from occurring by implementing various strategies and protocols as well as medical devices, but the prevention of these conditions are not easily achievable. To date, these conditions are still very much prevalent, and more efforts need to be invested to mitigate them.

#### 1.2 Importance of the Study

Hospitalization can be a stressful experience for many patients. After receiving a less than positive diagnosis that requires them to be hospitalised, these individuals are essentially putting their trust in medical professionals and the



healthcare system; trusting that they will be able to walk out of this place in a healthier state. In this manner, it could be said that HACs are not only dangerous in a physical sense, but also affect a patient mentally, adding on anxiety and further financial burden.

Falls are the second leading cause of injury in hospitals and make up the majority of reported incidents within hospitals. Though most incidents are mostly harmless, a bad fall on a weak or geriatric patient can severely affect an individual's ability to function. On the other hand, pressure injuries, otherwise known as bed sores, are caused by constant pressure, shear and friction on areas of skin and underlying tissues, commonly seen in patients who are bedridden and cannot easily move. It has notoriously slow healing rates and are a huge hindrance to an individual's day-to-day lifestyle. Hence, hospitals are advised to implement strategies to prevent these conditions from even occurring rather than working on treatment plans. Current preventive measures for both of these conditions come in a combination of medical devices and protocols.

### **1.3 Problem Statement**

Although incident rates of falls and pressure injuries have decreased over the years with the implementation of prevention strategies, they are still very much prevalent. Much research has been conducted over the years and the best strategies at present involve a number of different elements that mainly center around mitigation protocols conducted by nurses in the form of with regular rounding and repositioning. These strategies have presented several weaknesses, especially when there is a lack of staff. Some of these disadvantages include patient comfort and adherence in maintaining consistent performance.

The proposed project calls for the development of instrumented bed monitoring system that can detect one's presence and map their movements on the hospital bed. The continuous monitoring can potentially help increase the effectiveness of these protocols as nurses can be aware of any notable patient activity and act accordingly, where for example, if a patient has recently changed positions in their sleep, when the nurse comes along during rounds, they will know there is no need to disturb this patient.

#### **1.4 Aim and Objectives**

This project aims to develop an instrumented bed monitoring system that can aid in the prevention of unauthorised bed exits, falls and pressure injuries. Continuous monitoring of these patients makes sure that all important incidents are reported, aiding healthcare professionals to make a better-informed decision regarding the protocols to be implemented. It is hoped that with this device, some burden can be lifted from front-line healthcare workers.

The primary objective of this project is to develop an instrumented bed monitoring system that can provide continuous monitoring of a patient's movement on a hospital bed, allowing for timely interventions along with the documentation of any major patient activity. With this information, the frequency of manual patient checks can be reduced, effectively decreasing staff workload while promoting patient comfort and safety.

#### **1.5 Scope and Limitation of the Study**

Considering the aims of this project, the device is expected to have the functionality of a standard bed-exit alarm system while being able to map out one's laying posture on a bed. If a patient attempts to leave their bed without authorization, the healthcare providers will be alerted. Conversely, if a patient stays immobile for too long, alerts will also be given to indicate that repositioning is needed.

For this project, only a single bed layout has been considered. Patients or items that are too light may not trigger the sensing mechanism. Furthermore, due to the sensor design, consisting of an array of qualitative nodes, it is not possible to identify the highest points of pressure. Data logging and online notification may also be a challenge in places with poor network coverage.

## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 Introduction

Hospital beds are a vital part of patient care in medical facilities around the world. In this literature review, the prevalence, severity, and current prevention practices of bed-related hospital incidents, such as unauthorised bed exits, accidental falls, and pressure ulcers, shall be examined. Furthermore, studies on existing products, for example bed-exit alarms and smart hospital beds, will be conducted. Additionally, this section also discusses sensor designs that have been used to evaluate one's position on a bed.

#### 2.2 Prevalence of Bed-related Hospital Incidents

Hospitalized patients have a perennial relationship with hospital beds, where many cannot afford to take even a step away from the bed for the majority of their stay. This inevitably leads to a large number of bed-related incidents occurring in healthcare facilities worldwide. Bed-related HACs raise major concerns within the healthcare industry as they pose a significant threat to the patient's well-being and safety. As such, numerous studies have been conducted to ascertain the frequency and severity of these incidents, covered in the subsections below.

##### 2.2.1 Prevalence of Unauthorised Bed Exits

Being in a single place for extended amounts of time can take a toll on most individuals. This is especially true for those in a hospital setting. Although hospital inpatients are regarded as unwell, there is a good proportion of individuals that are hospitalized simply for further observation. In actuality, most inpatients still follow and function according to normal standards of time, where they are conscious for most of the day and sleep at night. Since there is not much entertainment to be found in a common ward, it is inevitable that some patients will occasionally leave the ward for unscheduled, non-clinical reasons, sometimes just to wander around the hospital. Oftentimes hospital staff would even recommend patients to walk around and take a breather from the ward.

However, this practice is advised to only be done either in the presence of family, caretaker or with the acknowledgement of the staff. Nevertheless, the majority of patients return to the ward or their own homes within two hours of leaving the ward.

The true risk of this situation lies with patients who wander off alone without anyone's knowledge, especially for those that are still unauthorised to leave their bed due to their condition. This condition may be physical or even psychological. Wandering alone is a huge safety risk to these populations who are currently weaker than their normal state, whereby those with physical injuries can potentially tear their wounds, end up stranded and miss prompt treatment. On the other hand, for those with psychological conditions such as Alzheimer's disease, wandering could potentially put them in harm's way as they are at elevated risks of getting lost and confused.

The more severe cases come in the form of absconders, people who decide to flee from their place of treatment, usually before the completion of their treatment plan. Numerous factors can contribute to absconding. This includes drug and alcohol addictions, intolerance of the hospital, fatigue, boredom, hopelessness, goal-directed motivations, behavioural disorders and more. However, the inability to pay for their medical expenses has been reported as one of the primary reason patients flee from the hospitals (Mahnaz Moradpour et al., 2021). Furthermore, research has shown that a larger proportion of these absconders are usually single males of younger age or married females, with underlying mental illness or addiction (Azadeh Memarian et al., 2015). Although most absconding events were reported to be of short durations, where most returned to the hospital on their own, this behaviour poses a certain risk of detrimental effects to the hospital, the community, and the absconders themselves.

### **2.2.2 Prevalence of Falls in a Hospital Setting**

Falls are one of the most common occurrences experienced by hospital inpatients, with about two percent of hospitalised patients falling during their time at the hospital. Among these falls about one in four will lead to an injury, whereby roughly 10% will result in a severe case (LeLaurin and Shorr, 2019). A considerable proportion of these severe injuries are seen in geriatric patients,

as they are subjected to deteriorating levels of function in their body such as poor eyesight, brittle bones and more. A study performed in a geriatric hospital reported 5.4 falls per one thousand days hospitalised, with a total of 1.3 percent leading to fractures (Menéndez et al., 2013).

Henceforth, falls ought to be taken seriously, as evidenced by the significant impact a post-fall syndrome can cause on the patient and their families. A single fall can further limit daily activities and is characterized by loss of autonomy, confusion, functional dependence, immobility, and depression. As a side note, the World Health Organization (WHO), estimates about 684000 deaths globally from falls, making it the second leading cause of unintentional injury and death.

There are several different estimates of falls in a hospital setting. This may be an effect of underreporting regarding light falls. In fact, the estimates are seen to range from about 20% of these falls happening from the bed, as reported by Healey and Scobie in 2007, to roughly 60% to 70% falling from the "bed or bedside chair," as claimed by Oliver in 2002 (Morse et al., 2015). Despite the variations in fall rates, it is clear that the bed is a substantial site of many patient falls. In fact, falls frequently occur when a patient is getting into or out of bed, when transferring to and from a wheelchair, reaching from the bed and even rolling out of bed.

This links to the subject discussed in the subsection above, whereby patients, who were not supposed to leave their bed, at that moment in time decide to do so without anyone's help. It is known that most hospitalised patients who were previously bounded to the bed for extended periods have weakened muscles. That is why they should be aided carefully on and off the hospital bed. However, many patients are eager to become independent as soon as they can, pushing themselves to do things without assistance. These actions put them at a higher risk of injury as they are likely to conceal their actions. One study even reports that the effects of falls in single-room patients were considerably more severe than those stationed in multi-bedded wards (Singh, Okeke and Edwards, 2015). This is likely due to the fact that they would be found later by nurses or companions in the event of a fall, effectively delaying treatment.

### 2.2.3 Prevalence of Pressure Injuries

Pressure Injuries, otherwise known as pressure ulcers, decubitus ulcers or bed sores, are a common problem experienced by patients who are confined to their beds for extended periods of time. When there is prolonged pressure exerted on a body part, such as the skin, it can cause localised tissue damage and pressure ulcers. Friction and shear forces are also contributing factors to this condition. It usually occurs on body prominences of the body, like the hips, tailbone, elbows and more. According to studies done in Europe, the median prevalence of pressure ulcers was 10.8%, with the most common site for pressure ulcers being the sacrum (Moore et al., 2020).

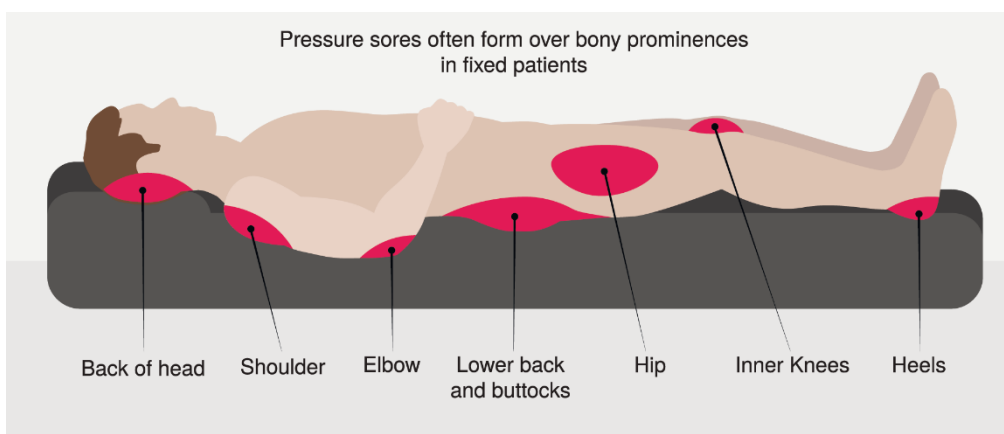


Figure 2.2.1: Common Sites for Pressure Sores (Xsensor.com, 2021)

Pressure ulcers are especially prone to developing in patients who are unable to move or change positions on their own, making this condition epidemic among geriatric patients as well as those who have limited motor functions. Notably, two-thirds of cases occur in patients over the age of 70 and depending on the type of healthcare environment, there have been reported to have different pressure ulcer prevalence rates. These rates range from 3-17% in non-surgical hospitalised patients, 17% to 28% in nursing homes and up to 66 % for hospitalised surgical patients (Bansal et al., 2005). It is likely as such because most surgical patients would be immobile for a period of time after surgery. In regard to nursing homes, some argue that there is a lack of continuity of care, whereby the quality of care is lower in nursing homes. According to a study in 2008, at the time of admission, the prevalence of pressure ulcers can amount to 26.2% among those admitted from a nursing home, while patients admitted from

other living arrangements were reported to be 4.8% (Eithne Keelaghan et al., 2008). In fact, immobility as little as of two hours is sufficient to create the basis of a decubitus ulcer for a bedridden or surgery patient.

This injury is categorised by the National Pressure Ulcer Advisory Panel (NPUAP) into 4 main categories. Stage 1 of a pressure injury comes in the form of non-blanchable erythema of intact skin, that may feel itchy and hot to the touch. In stage 2, the skin is broken, and a shallow wound or blister is present. From here, it can progress rapidly into stage 3 where full skin loss is seen, tunnelling of the tissue like a crater may occur. Stage 4 is diagnosed when there is full skin and tissue loss, whereby bone and ligaments can potentially be visible. Moreover, the incidence rates for the first, second, third, and fourth stages were reported to be 45%, 45%, 4%, and 4% respectively (Lotfolah Afzali Borojeny et al., 2020). Once past stage 2, it is very hard for the wound to ever heal completely. In addition, recurrence rates in the same spot can be as high as 90% (Bansal et al., 2005).

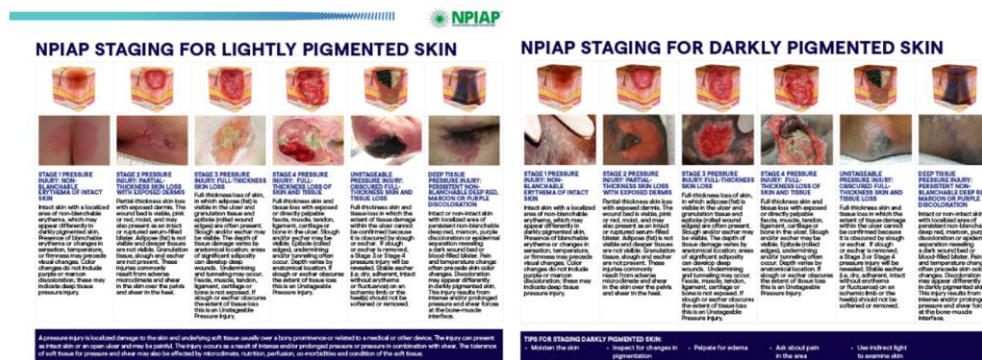


Figure 2.2.2: NPIAP Staging for Pressure Sores (Npiap.com, 2019)

Furthermore, pressure injuries are likely to come with deadly complications such as infections. Since the skin has been compromised, the microbes can freely enter through any stage 2, 3 or 4 bed sore. It goes without saying that this debilitating condition puts a lot of burden on the affected individuals, the healthcare system and social economic costs. According to the National Pressure Ulcer Advisory Panel (NPUAP), pressure ulcers affect about 2.5 million patients annually in the US, and the cost of management and treatment can run anywhere between \$9.1 and \$11.6 billion (Ahrq.gov, 2014).

## **2.3 Current Prevention Measures**

These issues have been plaguing the healthcare industry for a while now and in order to minimize the frequency of these incidents, healthcare providers have introduced a number of preventive measures. This includes protocols such as regular patient assessments and staff training, as well as devices and products like smart beds, bed-exit alarms, bed rails and more. This subsection will explore the current prevention practices and their effectiveness in reducing these incidents.

### **2.3.1 Current Prevention Practices for Combatting Pressure Injuries**

Pressure ulcers are painful, difficult to treat and can worsen rapidly if not managed. There is a high rate of recurrence, and severe cases may never heal completely even with surgical intervention. This is why more efforts have been put forth into preventing bed sores from occurring. The SSKIN bundle – skin, surface, keep moving, incontinence and nutrition, is an aid used in planning preventive measures for pressure injuries.

The skin is the largest organ of the human body and main victim of pressure injuries. Regular skin assessments to check the skin for any signs of discolouration, localised heat, redness, oedema, or breakdown acts as an early detection measure for bed sores. In fact, these risk assessments for pressure injuries should be carried out at the time of admission, identifying the skin fragility of the patient. Incontinence, in particular, can aid in the breakdown of skin. This is why any external moisture, including sweat or wet skin, should be wiped dry as much as possible. Of course, proper nutrition and hydration also plays a role in preventing the formation of bed sores. Adequate vitamins, minerals, proteins and fluid intake helps to keep healthy and promotes healing. (Nice.org.uk, 2014)

Another main factor in the SSKIN bundle is to keep moving, whereby immobilisation should be prevented from lasting for more than two hours. While patients are encouraged to do so themselves, nurses will assist bedridden patients to reposition every two hours to six hours depending on the protocol. Additionally, patients should not be repositioned onto reddened areas, as it is an indicator that the tissue has not recovered from previous pressure loadings (British Journal of Nursing, 2020). The NPUAP recommends 30° tilted side-



lying positions, alternating from the right side, back and then back to the left side. Prone positions can also be considered.

Since pressure ulcers form from continued laying on beds, it is acknowledged that the surface plays a major role in the formation of bed sores. Multiple special mattresses and cushions have been designed over the years. A prominent product is an air or fluid-filled overlay. These beds can be adjusted to distribute the patient's weight and provide extra support where needed. According to some studies, mattress overlays with pressure-relieving features are more effective than a standard mattress or a foam mattress at preventing pressure ulcers in nursing homes and intensive care units (Brecht Serraes et al., 2018). While there is not much evidence on whether a static or alternating overlay air mattress is more effective, at the moment alternating overlays are more widely used.



Figure 2.3.1: Pressure Relieving Air Mattresses: Static (Left), Alternating (Right) (Health, 2023; Air Mattress Overlay, 2022)

However, it is important to note that these interventions must be implemented together as a set. Previous research has shown that single interventions are not effective in preventing bed sores, hence the SSKIN bundle. At present, the largest obstacle is faced is ensuring that consistent repositioning is taking place as there is a lack of nursing staff worldwide.

### 2.3.2 Bed-Exit & Fall Prevention Measures

As mentioned before, incidents involving bed-exits and falls within a hospital are often intertwined, and while there are some separate practices designed specifically to target each issue, most preventive measures incorporated deal with both bed-exits and falls.

One main practice being used to counter fall hazards is the fall risk identification and assessment. Popular tools and assessment techniques, with

acceptable specificity and sensitivity are for example: the Hendrich II Fall Risk Model, The Morse Fall Scale, and the STRATIFY tool. These assessments evaluate patients on their likelihood of falling while they are at the facility, whereby the goal is to proactively identify patients who may be at high risk of falling and implement tailored interventions to lower that risk. Various factors that contribute to the possibility of falling are taken into consideration such as the patient's medical history, current medication regime, mental status, and mobility. However, it is important to note that there is no gold standard for risk assessments and these three tools are not all-encompassing and cannot predict all falls (Callis, 2016).

With regards to the true effectiveness of these tools, some research has reported that the accuracy of these tools is only as, if not less, effective than a nurse's clinical judgement (Oliver, 2008). Therefore, hospitals are implementing purposeful rounding as frequently as on an hourly basis. This practice has brought forth positive effects on patient care but has its own set of challenges presented by front-line staff. The sustainability of the protocol has been questioned as it causes the staff to have an increased workload and competing priorities. Additionally, it has been proclaimed that the adherence to rounding is often not consistent. This may be improved with proper support and push from healthcare administrators but requires much work to be put in by the hospital leadership. (Toole, Meluskey and Hall, 2016).

Sitters, otherwise known as 'specials', is another rather costly strategy. These caretakers will act as a companion and will provide continuous surveillance of the patient. While there is some evidence suggesting the effectiveness of sitters, other studies have shown that fall rates remained the same even with the presence of sitters. (Wood et al., 2018)

Other than these protocols, environmental modifications can be done to help mitigate the number of falls in the hospital. This includes lower bed height, bedside rails, and the use of non-slip socks. Higher bed heights were initially thought to be an increased risk factor for patient falls both due to patients rolling off the beds at a larger distance to the floor as well as the difficulties experienced by shorter patients when getting off the bed. Yet, new research has shown that particularly low beds were also dangerous. This was because of uncontrolled descents onto the bed due to its unexpectedly low height.

This meant that ideal bed height varied from patient to patient and would differ depending on that population's average. (Morse et al., 2015).






Above all, the use of bed-exit alarms is arguably one of the most effective technological advancements in regard to patient wandering and accidental falls. These alarms help alert caregivers when a patient attempts to get out of bed, where presumably the caregivers would have enough time to offer assistance before a fall can occur. There are various types of bed-exit alarms available on the market, ranging from pressure-sensitive pads to garment clipping sensors, infrared lasers and cameras.

By far, pressure-sensitive pads seem to be the most popular form of bed-exit alarms, which are found in both independent and smart hospital beds with built-in alarm systems. Table 2.1 below provides an overview of pressure mat-based independent alarm systems, where it can be seen that all models have similar-sized pressure mats at an average of 77.4 by 35.3 cm. These pressure mats are usually placed either at the buttocks, shoulders or thoracic area. A major limitation is presented as false alarms, whereby the patient simply is not within the area of the pressure sensitive mat, despite still being on the hospital bed. When it comes to smart hospital beds, several major companies such as Hill-Rom, Stryker and Linet have incorporated bed-exit alarms into their catalogue of hospital beds. In particular, Stryker has a 'Chaperone' bed exit alarm system, consisting of three functional zone settings, with the largest coverage setting being able to cover the entire bed.



Figure 2.3.2: Stryker Chaperone Bed Exit System (S3 ® MedSurg Bed, n.d.)

Table 2.3.1: Comparison of Pressure-Mats Bed-Exit Alarm Systems (Mat, 2018; Amazon.com, 2023; Medpage-ltd.com, 2020; Malaysia)

Bed-Exit Alarms	TumbleCare	Lunderg	Alerta	Smart Caregiver	Secure Safety Solutions
					
Mat Size (cm) [Length x Width]	75 x 45	83.8 x 25.4	76 x 25	76.2 x 50.8	76.2 x 30.5
Alarm indicators	Flashing LED with 3 levels of alarm tone	Vibration and controllable volume	Controllable volume and tone	Controllable volume and tone	Flashing LED with 80dB alarm
Wireless Alarm Module	YES	YES	NO	YES	NO

## 2.4 Sensor Designs

Within reported literature, there are many different takes on how sensor systems, aimed at measuring human movement, can be designed to lay flat, flush with the mattress. Some of these studies have been specified to be tackling pressure injuries or bed-exits, while others were extracted from literature meant for sleep analysis.

In Malaysia, a M-BAS sensor system, meant for bed-exits, has been developed and trialed at the geriatric ward of UMMC. This sensor is unique in its modular design, that consists of three main panels placed across the bed's width. The center panel functions to silence the alarm when pressure is detected. Therefore, when the presence of pressure on either of the two side panels being absent in the center panel, the patient is assumed to be attempting to get off the bed and an alarm will be triggered (Kogilavani Subermaniam et al., 2017).

Another research paper aiming to solve pressure ulcers, developed a motion detection system with four force-sensing resistors and five accelerometers. These sensors were fixed to the bed on a specialized belt system, that arranged them in a configuration that covered both the top and bottom of the mattress, whereby four accelerometers were in each corner of the top side of the bed, while the remaining one was placed in the center on the bottom side (Hayn et al., 2015). They also paid attention to the possibility of incontinence and isolated the sensors with shrink tubes.

A study conducted in 2020 on developing a smart bed for sleep disorders, made use of a pressure mapping system. Their design could detect the subject's laying pressure distribution, which allowed them to deduce the movements, positions and breathing rate of their subject. The design consisted of three layers. The central layer was made of pressure-sensing piezoresistive fabric with two outer fabric layers sandwiching it had integrated row and column conductors (Laurino et al., 2020). Accelerometers were also utilized to measure the ballistocardiograph (BCG) signal.

Although more literature was browsed through during the course of research, these three papers in particular impacted how this project's design was planned.

## 2.5 Summary

In short, falls, unauthorized bed exits, and pressure injuries are still a concerning issue to many hospitals. They bring forth lots of complications and are a pain to treat. This is why most hospitals worldwide have turned to implementing prevention measures in efforts to mitigate these HACs. However, it is easier said than done. Current practices involve a combination of various interventions that include staff protocol and technological assistive devices. Through this review, it was found that these practices revolved around regular rounding by medical staff. This is seen in pressure injuries where even with the implementation of air mattresses, regular repositioning by staff still had to continue. This presented a key weakness whereby there is a lack of staff, training and most importantly adherence to their duties. On the positive side, bed-exit alarms in the form of pressure mats, independent or integrated into smart hospital beds, have proven a certain effectiveness.

Given the similarities, this project aims to develop an instrumented bed monitoring system that can function as a bed-exit alarm and contribute towards mitigating pressure injuries. After reviewing several studies focusing on sensor design for bed monitoring, considerations and tweaks were made to the sensor design based on their achievements and discoveries. More details will be explored in the next chapter

## CHAPTER 3

### METHODOLOGY AND WORK PLAN

#### 3.1 Introduction

This chapter will outline the methodology and work plan for the development of an instrumented bed monitoring system that aids in preventing falls and pressure injuries in the healthcare settings. The report will describe the different stages of the device development process and reasoning behind each decision.

#### 3.2 Selection of Main Components

Selecting the right components is a crucial step in any system design, making the right compromises when comparing between a component's specifications, cost, durability, availability on the market, and compatibility with the system.

##### 3.2.1 ESP32

The ESP32 is a low power, low cost, versatile microcontroller board by Espressif System that has become increasingly popular in the recent years. This board is based on Espressif ESP32 system-on-a-chip (SoC) and serves as the successor to the ESP8266 boards series. The SoC is a powerful chip that contains a dual-core processor that is easily programmable with many IDEs, including the highly popular Arduino IDE.

For one, its connectivity options are one of their main redeeming features. With built-in Wi-Fi and Bluetooth, the ESP32 board can easily connect to a variety of devices and networks, making it a suitable choice for projects that require wireless communication. such as IoT devices. On top of that, the ESP 32 has a pin multiplexing feature, allowing several peripherals to share a single GPIO pin. For this particular project, the ESP-WROOM-32 development board was chosen.

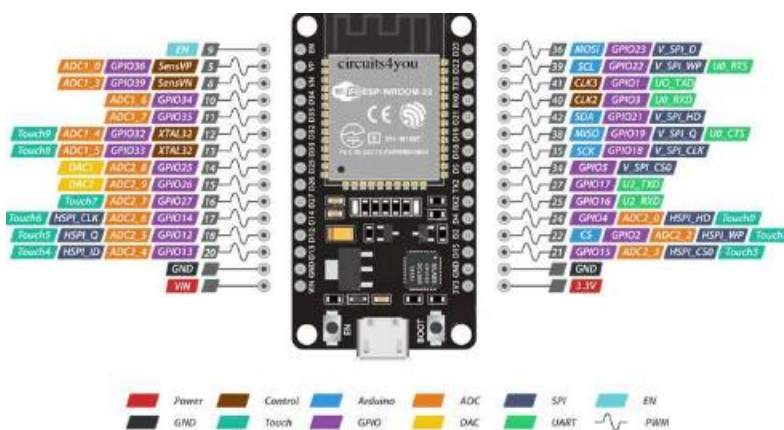


Figure 3.2.1: ESP-WROOM-32 Board Pinout Diagram

(Lastminuteengineers.com, 2023)

### 3.2.2 I/O Expanders

Although the ESP32 has a considerable number of GPIO pins, this project will incorporate a large sensor array big enough to cover a single bed - more details on the sensor design will be covered in the subsequent subchapters; hence I/O expanders will be included in this project.

A total of two I/O expanders will be utilised in this project. They are the MCP23008 and MCP23017, which provides 8 and 16 I/O slots, respectively. These I/O expanders are by Microchip Technology and are fairly popular components that can be easily sourced. They have low power consumption and can operate at voltages ranging from 1.8V to 5.5V, making them compatible with the ESP32 which runs at the voltage of 3.3V. The I2C interface will be used for communication between the ESP32 and these I/O expanders, resulting in only four main wires, two from each I/O expander to be connecting to the ESP32. This frees up the other pins on the ESP32 to be used for other functions.

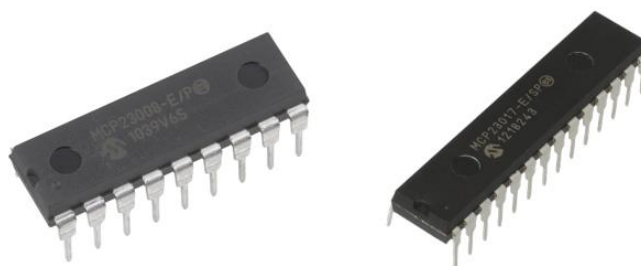


Figure 3.2.2: I/O Expanders - MCP23008 (left) and MCP23017 (right)



### 3.2.3 Touchscreen

In the ever-growing digital era of smartphones, the use of touchscreens has become progressively widespread. This is because a touchscreen provides an enhanced user experience thanks to its intuitive and highly interactive user interface. It is also largely visual-based, which falls into play as humans tend to absorb information better and faster with the help of visual stimuli. Even though touchscreens have not really made their mark among student projects due to their high prices, the advantages they bring to the table are undeniable.

Since this project involves the mapping of one's position on the bed, the common LCD options such as the 16x2 and even the 20x4, are unsuitable as they are text based and do not have enough space to display the required pixels. Other cheap non-touch graphical display options such as the OLED LCD, often come in sizes of 0.96 or 1.5 inches, which is still too small for a comfortable user experience. While there are other types of non-touch graphical screens available on the market, most are priced at similar price points to a touchscreen, without the benefits brought along by touchscreens.

It is also important to note that touchscreens were less popular for student projects due to the complexity in programming them. Hence for this project, a touchscreen with TTL serial input was selected – the Nextion Intelligent NX8048P050-011C HMI capacitive display. In comparison, there are other TTL serial input displays from 4D Systems and Waveshare, however Nextion has their own IDE software that allows easy development of user interface. This is a powerful model that has its own onboard processor, letting it perform a variety of functions including playing music files from an SD Card.



Figure 3.2.3: Nextion Intelligent NX8048P050-011C HMI Capacitive Display

(Zhan, 2022)

### **3.3 Sensor Design**

Another indispensable part of this project is the sensor design. Multiple sensors are needed to accurately detect the movements of a patient on the hospital bed. Throughout the process of designing, numerous factors were considered and will be discussed in the detail in the following subsections.

#### **3.3.1 Commercial Grade Sensors**

There are many types of sensors that can be used measure the movement of a human being. Among them, accelerometers and gyroscopes are usually implemented in wearable devices, such as virtual reality gaming consoles and smart watches. Other sensors such as ultrasonic and infrared motion sensors rely on reflected waves that bounce off the body to determine the body's position. These two categories of sensors, in particular, were determined to be less suitable for this project, as they would not be able to bring forth the desired results. For one, due to patient comfort and privacy issues, this project will not be encompassing wearable features that can potentially track down a patient. On the other hand, ultrasonic or IR sensors cannot accurately detect minute movements such as tossing around on the bed. They also have to be strategically placed to receive all the required reflected waves. This meant that these sensors might have to be mounted directly above their targets, often on ceilings or walls, which can make installation complicated and harder to replicate.

For this instrumented bed monitoring system, it was decided that the detection would come from the direction of the bed itself, whereby the sensor layer could be simply placed onto of any mattress to start functioning. The idea was to replicate a mattress protector, whereby the user would not even realise that there was such a layer beneath them. This meant that the design had to supply as little discomfort as possible to the user.

Taking that into account, the multiple types of thin, flat sensors were extensively explored. The flatness of the sensor was an important feature taken into consideration, as it would play a part in allowing the user to comfortably sleep on top of this sensor layer. This narrowed the scope down to three potential options which were thin film pressure sensors, velostats, and flat strain gauge sensors. All of these sensors are made from flexible, thin materials that can be used for similar applications. They function by measuring the change in

resistance or capacitance obtained during deformation when there is a force or pressure applied to it. When put in comparison, thin film pressure sensors and flat strain gauge sensors have comparable sensitivity and accuracy, with velostats, lagging slightly behind. Moreover, thin-film pressure sensors and velostats are relatively expensive, coming up to above RM 20 per sensor. Flat gauge pressure sensors, such as the ones in the lower right of Figure 3.3.1, are considerably cheaper.

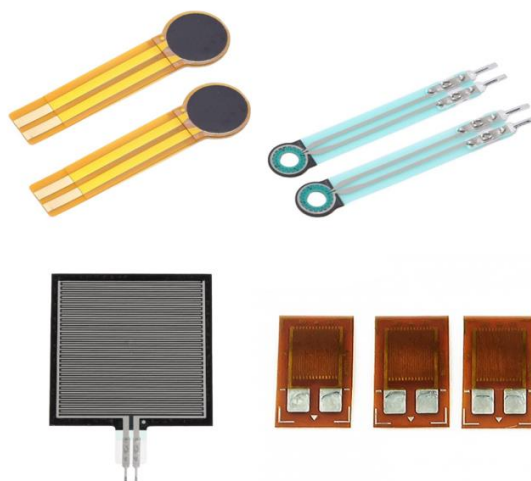


Figure 3.3.1: Examples of Thin-film Sensors and Flat Strain Gauge Sensors

All of these commercial sensors listed above would require two wire lines from each sensor. If every sensor placed is set as a single node in a sensor array, this meant that the number of wires exiting the bed would amount to double the number of nodes designed for the system. In order to better map out a user's sleeping position on the bed, it is understood that more nodes would yield more accurate results. However, the cost in buying many of these commercial sensors along with the insane number of wires that would be required to be attached to them limited the number of nodes that could be placed into the design.

Hence, the design was eventually settled on a self-constructed sensor array, where instead of a single layer of single node pressure sensors, the array would consist of two layers, one with vertical running lines and the other with horizontal running lines. This made the wire management easier as the total number of wires would simply be the sum of the number of rows and columns in the sensor array.

### 3.3.2 Design of the Sensor Array

As mentioned earlier, this self-constructed sensor array would consist of two sensor layers. The top layer would comprise of horizontal sensor lines, running across the bed from left to right, while the bottom layer contained vertical sensor line, running from the top to the bottom of the bed. This is demonstrated in Figure 3.3.2, whereby the yellow lines represented the top layer, and the blue, the bottom layer. Thus, when there is a pressure applied onto it, the two layers would touch and produce a signal which can be read by the microcontroller, indicating that there is something at a particular location of the sensor array.

The weakness that came with this design is that the results recorded would be qualitative instead of quantitative, whereby the true pressure exerted cannot be measured by this system. However, this design allows for a larger number of sensor nodes, which in turn leads to more accurate mapping of the user's laying position on the bed.

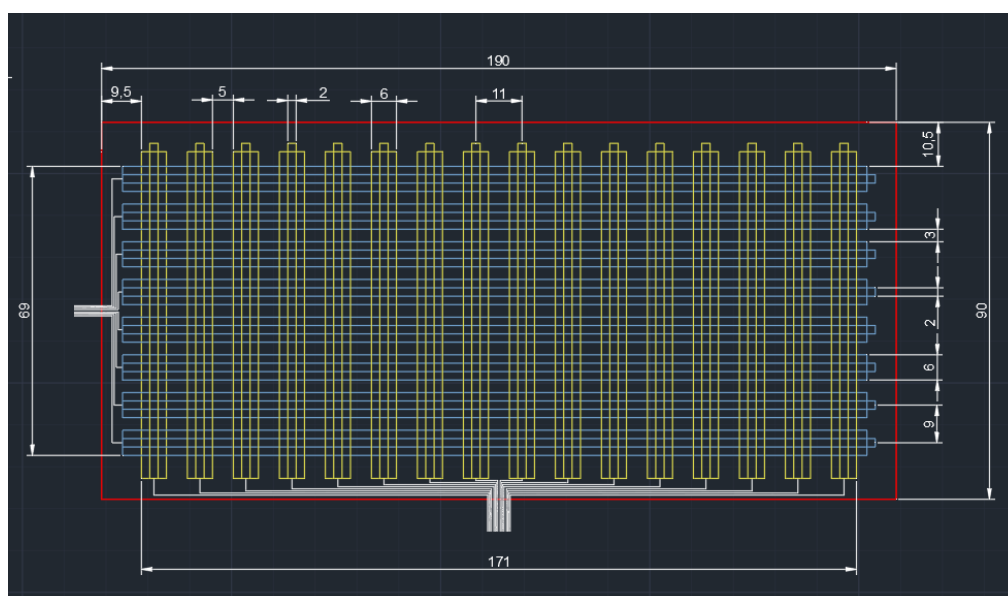


Figure 3.3.2: Bed Array Sensor Design on a Single Bed

The number of sensor nodes was ultimately decided to be 128 in total. This was obtained by having eight vertical lines and sixteen horizontal lines evenly distributed in our array and was decided upon in order to maximise the use of the 2 I/O expanders described earlier. This same circuit design would only be able to support twenty-four commercial pressure sensors compared to the 128 nodes in this array. It is important to note that, at current, only a single bed layout, measuring 190 cm by 90cm, has been considered for this project.

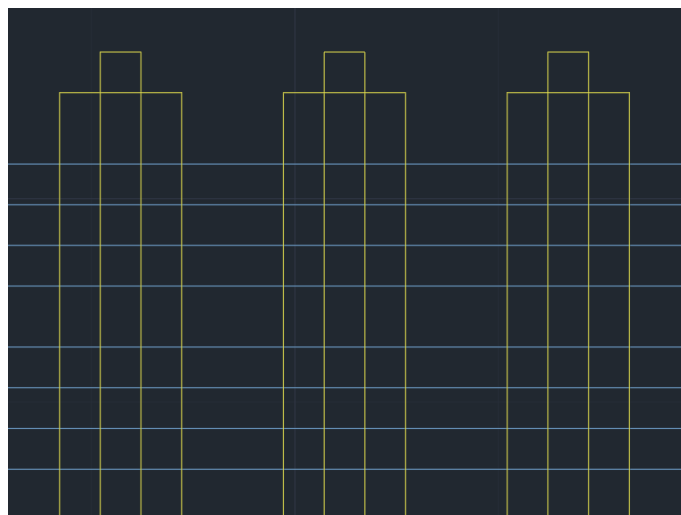


Figure 3.3.3: Close-up of Sensor Design

With some consideration, it was decided that each line would consist of two loops. This allowed each sensor node to have sixteen potential contact points, decreasing the possibility of not picking up a pressure signal. For this single bed configuration, the gaps in between the nodes are 5cm by 3cm, which a conservatively small area that can afford to be ignored when the target is a human with an average size of 1.6m by 30cm. Moreover, the two loops of each line intersect one another with the inner loop extending out further than the outer one, ensuring that the signal could still return to the input pins to be detected in case any of one of the lines could have ended up broken.

Initially, this design was constructed with anti-static threads in mind. These threads were often used for anti-static textiles, such as gloves, jackets and more. They are made up of a stainless-steel fibre wound together with polyester fibres and can conduct electricity. It brought forward many benefits, such as making the entire sensor layer wash-proof and can be easily produced with the help of a sewing machine. However, actual testing of this material proved its inadequacy. Although the electrical conductivity was continuous throughout a long stretch of thread, the resistance it produced was in the Mega Ohm range. This made the current too small, cutting it close to the leakage current of the I/O expanders. Hence, this idea had to be scrapped and a new material had to be selected. The main choices at this time were either copper wires or stainless-steel wires. Velostats were also considered, however they were expensive, and much work would have to be done to set them in thin strips, while maintaining

continuity. Both copper and stainless-steel wires had incredible conductivity, but the stainless steel was eventually chosen as they could be easily obtained in an exposed form without any covering compared to copper wires. In addition, they could also be washed and had anti-rust properties.

Considering the large area to be covered, it was hoped that the sewing machine to continue being of use for this project and because normal cloth threads had diameters ranging from 0.05mm to 0.24mm, 0.2mm thick stainless-steel wire was selected. Figure 3.3.4 below displays a working prototype of the design with 0.2mm stainless steel wires, whereby it would be folded into two and pressed upon.



Figure 3.3.4: Prototype of a Single Sensor Node with 0.2 mm Stainless Steel

Although this prototype can be considered a successful proof of concept, there were several challenges faced in the course of developing this prototype. One major issue involved the construction of the prototype itself. As mentioned before, the current sensor design utilises 0.2mm stainless steel wires. It was originally chosen as it was on the larger side, to account for more surface area during sensing, while still remaining within the range of a normal thread thickness. However, in hindsight, the compressibility of normal clothing threads was not considered. Stainless steel, on the other hand, is obviously much less compressible. Although successful, this made the sewing process a little arduous, because the machine had to be tuned to higher levels of tension. The side effect of this situation was that the cloths used for the prototype easily crumpled during the sewing process if one was not being careful. This led to research and testing done on several different types of cloth material, including felt, cotton, velvet

and even jeans. Evidently, sturdier materials such as velvet or jeans held up better. However, this was not a fool-proof solution to this matter. Hence, 0.1mm and 0.15mm stainless steel wires were also tested for this sensor array. Their thinner diameter proved to exert less stress on the sewing machine with less visible crumpling on the cloth.

Eventually, 0.15mm stainless steel wires were selected for this project. The 0.1mm stainless steel wires were eliminated due to its weaker profile, whereby it had snapped a few times through the process of sewing. Therefore, the 0.15mm wires strike a balance between the effects seen with 0.2 and 0.1mm wires. It was stronger with a slightly larger diameter for larger surface area for touching, increasing the possibility of detecting presence, while exerting a more acceptable stress on the cloth and sewing machine.



Figure 3.3.5: VT8 Prototype Sensor Layer: Eight vertical sensor lines.



Figure 3.3.6: HZ16 Prototype Sensor Layer: Sixteen horizontal sensor lines.

Figures 3.3.5 and 3.3.6 above portray the actual single-bed prototype sewn for this project. Normal cotton cloth and 0.15mm stainless steel wires have been utilised. The yellow layer contains eight vertically running sensor lines, while the blue layer has sixteen horizontally running sensor lines. These two pieces are layered on top of one another right below the bed sheet and are fastened to the bed frame via Velcro. This is done by attaching dual-utility straps made of elastic and Velcro loop to the sensor layers, allowing a secure constant tension to be exerted on the cloth layers.

### **3.4 Electronic Design & Schematic**

The electronic design is a critical part of every device as it defines how a device would be programmed to perform its tasks. This project will be utilizing the main components outlined in previous sections, as well as other fundamental electronic components to form a fully functional electronic system.

#### **3.4.1 Printed Circuit Boards**

For this project prototype, the plan is to make and develop printed circuit boards (PCB). The PCB is a flat board made out of non-conductive material, such as fiberglass or plastic, with conductive pathways etched onto it. These pathways act as the wires in the circuit and are in essence the remnants a copper layer that has been washed off with acid on areas that should not be electrically connected. This decision was motivated, in part, by the author's own curiosity in learning how to create a PCB. The use of PCBs brings forth a number of benefits to the project, including better compactness, reliability, and stability of the circuit. This is due to the fact that the PCB is a more permanent form of circuitry compared to breadboards, which eliminates issues such as loose electrical contacts and dangling wires.

In recent years, PCBA is also becoming more readily available. Printed Circuit Board Assembly, otherwise known as PCBA, refers to the process of assembling electronic components, soldering them onto a PCB and testing it thoroughly to ensure that the circuit is working as it should. Companies are now offering these services to even low-volume PCB production, which makes it a great avenue for hobbyists and students. This service reduces the hassle sourcing



multiple electronic components from various vendors and saves the time that would have been spent on assembling the circuit.

### 3.4.2 Device Design and Layout

A total of three boards will be produced for this project: a main board and two peripheral boards, whereby one controls the vertical lines of the sensor array (VT8) and the other deals with the horizontally running sensor lines (HZ16). These peripheral boards are referred to as VT8 and HZ16 correspondingly because VT8 will have a MCP23008 connected to a total of eight vertical running sensor lines while the HZ16 has a sum of sixteen horizontal lines with an MCP23017. A small casing containing the main board, touchscreen, and speaker will be mounted to the headboard of the bed, while the peripheral boards will be positioned at the middle of the width and length of the bed as shown in Figure 3.4.1.

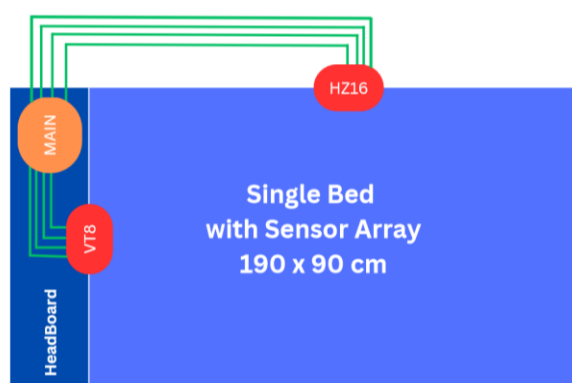


Figure 3.4.1: Device Set-up on a Single-bed Configuration

This design was implemented in an attempt to minimize the use of lengthy wires as much as possible, whereby each peripheral board would have four wires connecting to the main board: power, ground and two communication wires. This approach ensured that the many sensor columns and rows, totalling up to twenty-four lines, did not need to be wired all the way up to the main PCB at the headboard. Additionally, this consideration could potentially lessen the likelihood of noise interference, signal attenuation and data loss.

### 3.4.3 Circuitry Design & Schematic

Figure 3.4.2 below portrays the electronic schematic connections to the ESP32, which is the main microcontroller board for this project. Here, pins 2 and 5 will act as the SCL and SDA pins, connecting to both MCP23008 and MCP23017 on the VT8 and HZ16 boards respectively. They will communicate via the I2C protocol, where the ESP32 acts as the master and the two I/O expanders as the slaves with their own corresponding addresses. Since I2C bus drivers are ‘open drain’, a pull up resistor of 3000 Ohms is connected to keep the signal as normally high when there is no device trying to pull it low.

Pin 1 is the reset pin, and in this circuit, a push button to ground is attached to it. Moving on, pin 13 is connected to the buzzer via a 10uF capacitor. As a side note, an actual speaker will also be connected to the system via the touchscreen which is in charge of controlling it. Other than that, pins 3 and 4 act as the serial TTL pins which is linked to the touchscreen connector labelled LCD\_CN in Figure 3.4.2.

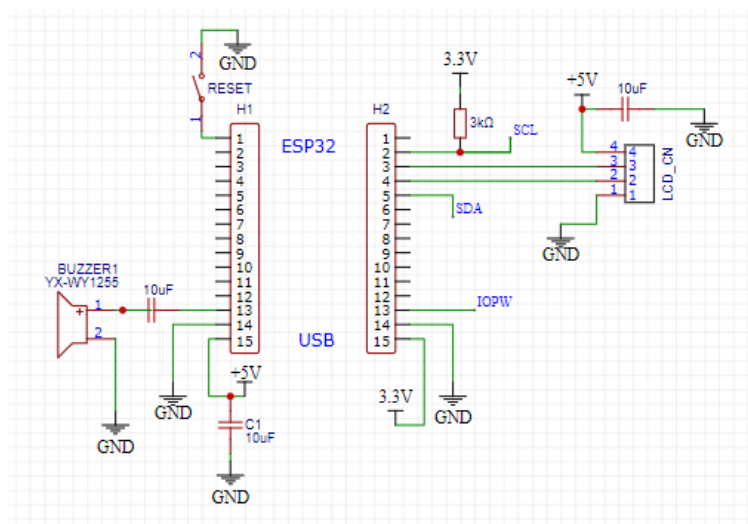


Figure 3.4.2: Schematic Connections to Main Microcontroller - ESP32

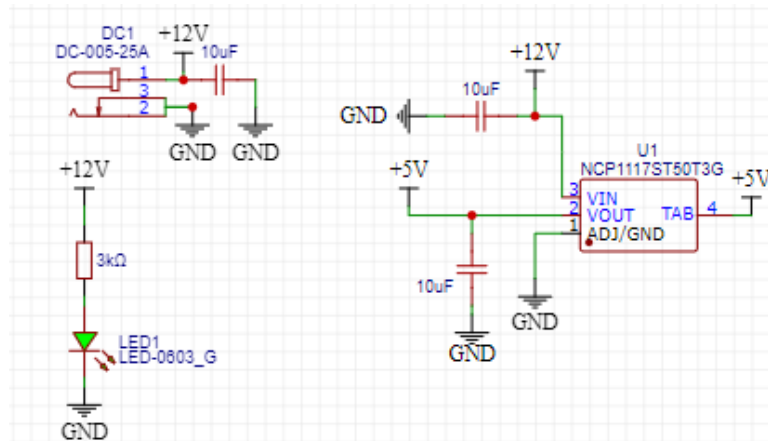


Figure 3.4.3: Schematic Circuitry for 12V Power Socket, LED & 5V Regulator

This device will be making use of a common 12V power adapter, which will be stepped down to 5 volts with the help of a NCP1117 series regulator by onsemi. Multiple 10uF capacitors are connected in parallel to filter out the DC supply and eliminate any AC ripples. A power LED is also included in this circuit to double check if power is being supplied to the circuit as shown in Figure 3.4.3 above.

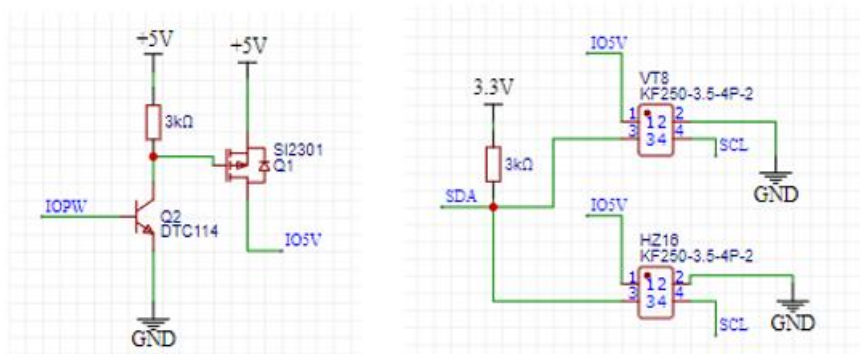


Figure 3.4.4: Schematic Connections to Peripheral Boards

As mentioned earlier, a pull-up resistor is attached to the SDA and SCL in order to restore the line to high when the device is idle. 3000Ohm resistors were used, whereby the pull up resistor on the SDA line can be seen in Figure 3.4.4, while the SCL's pull up resistor is shown in Figure 3.4.2.

Next, it is important to note that the power supply for the VT8 and HZ16 boards are provided by the main board. In the schematic diagram, this power supply is labelled as IO5V, which is controlled with a BJT MOSFET combo. This means that the main board will be able to easily reset the peripheral

boards by simply turning off their power source. This can be done by altering the output of pin13 on the ESP32, labelled IOPW.

Figure 3.4.5 shows a portion of the electronic schematic seen on the VT8 printed circuit board. A total of four wires come from the main PCB, namely power, ground SDA and SCL. The same is true of the HZ16 printed circuit board seen in Figure 3.4.6. It can be observed that the I/O pins on both MCP23008 and MCP23017 have been maximised, whereby 8 pins will correspond to the eight sensor lines in the vertical array and 16 pins for the sixteen horizontally running sensor lines. Each board also has their own power LED to verify that power is being provided.

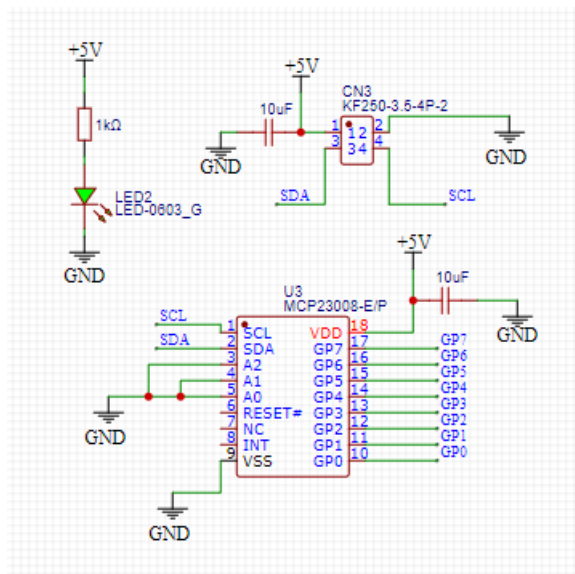


Figure 3.4.5: Schematic Connections to MCP23008 on VT8

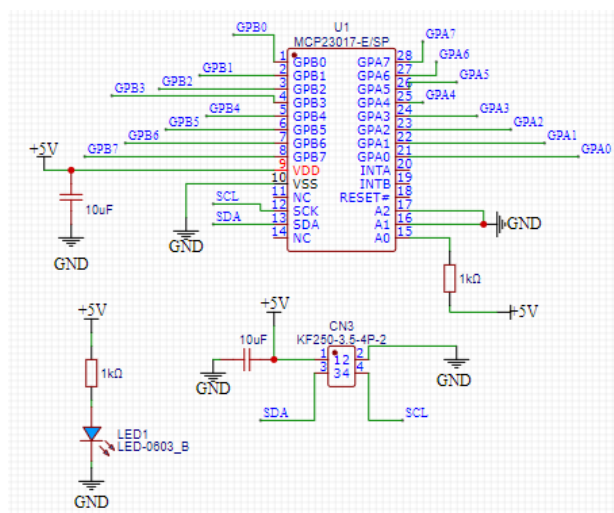


Figure 3.4.6: Schematic Connections to MCP23017 on HZ16

The remaining portion of the peripheral boards are the connections to the sensor array. In Figure 3.4.7, the connections of V8 are shown. It is observed that there is a pull up resistor of 12k Ohms at every input. On the other hand, schematic connections to the sensor array of HZ16 are directly linked to the pins of the MCP23017 as seen in Figure 3.4.8. This is the electronic design done in consideration of the reporting process, where the MCP23017 calls the horizontal sensor lines in turns, pulling down each line in a rolling manner. The MCP23008 in this case is the main receiver, recording which of the 8 vertical sensor lines provides a low output, indicating that pressure is being exerted at that node. Thus, allowing for the mapping of someone's position on the bed.

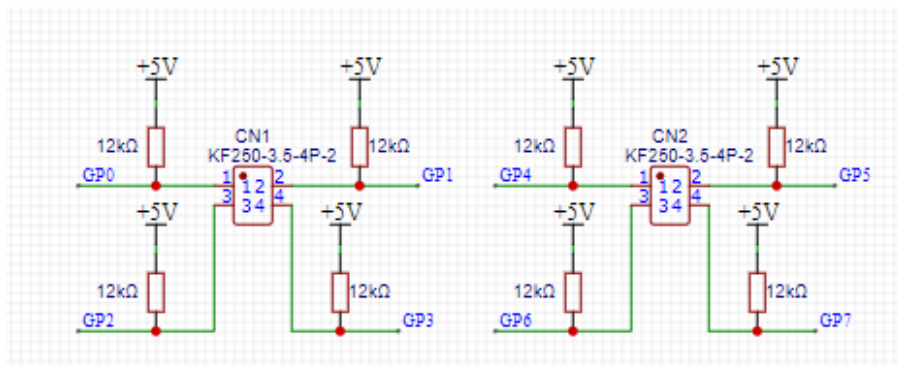


Figure 3.4.7: Schematic Connections of VT8 to Vertical Sensor Array

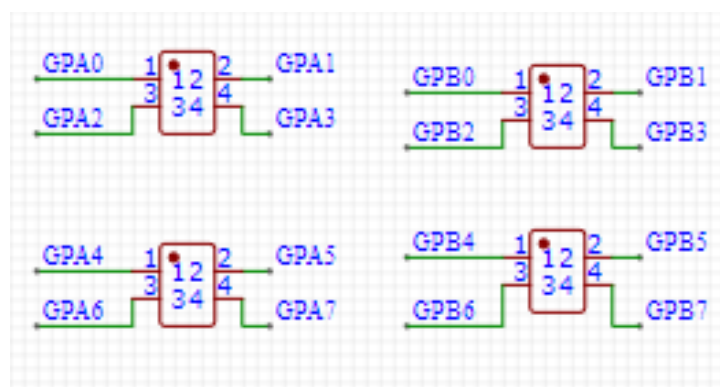


Figure 3.4.8: Schematic Connections of HZ16 to the Horizontal Sensor Array

### 3.4.4 PCB Design

This section outlines the elements taken into consideration during the process of designing the PCB boards for this project. Moreover, due to the price package for these PCBs, all components will only be present on the top side of the PCB.

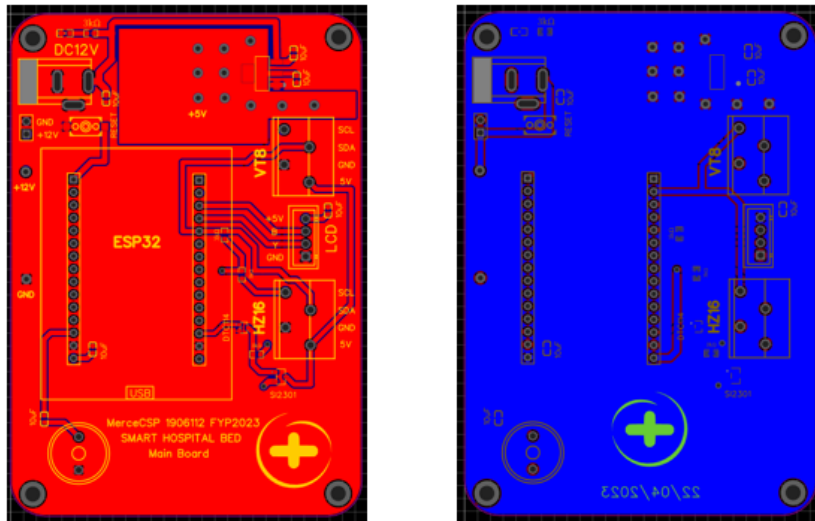


Figure 3.4.9: PCB for Main Board: Front View (Left), Back View (Right)

For the main board, the outline of the actual dimensions of the ESP32 chip has been added in the process of designing in order to prevent any crossovers with tall components, which can hinder proper insertion of the ESP32 chip into the pin header sockets. A makeshift heatsink was also added around the power regulator, in the form of a copper pour and scattered vias. It is anticipated that dissipation of heat may be needed due to the relatively large amounts of current consumed by the touchscreen.

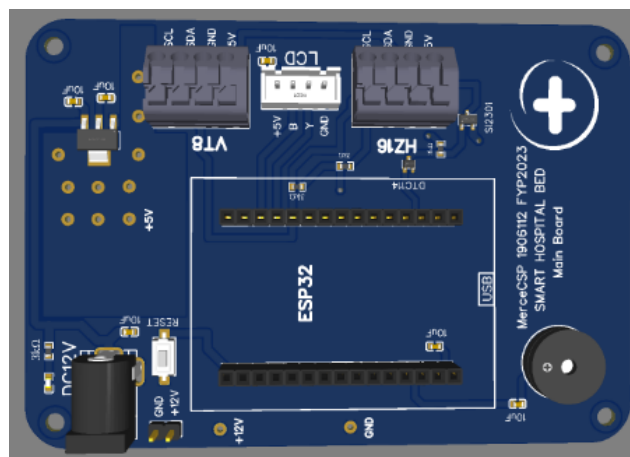


Figure 3.4.10: 3D-View of Main Board

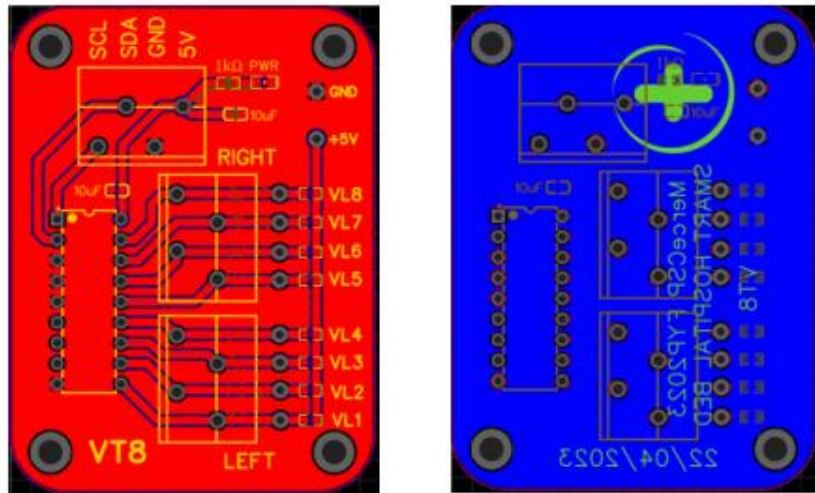


Figure 3.4.11: PCB for VT8 Board: Front View (Left), Back View (Right)

Figure 3.4.11 and 3.4.12 present the PCB designs of the VT8 and HZ16 boards respectively. Here, an exposed via was added to each outgoing sensor line. This was done to make measuring the voltages more easily with a multimeter. The same was done for power and ground of each board, whereby there would be a specific exposed via placed into the board simply for measuring purposes. Additionally, all three boards were designed with curved edges and holes in each of the four corners. This makes the boards easier to mount, if needed, by having an adequately sized screw.

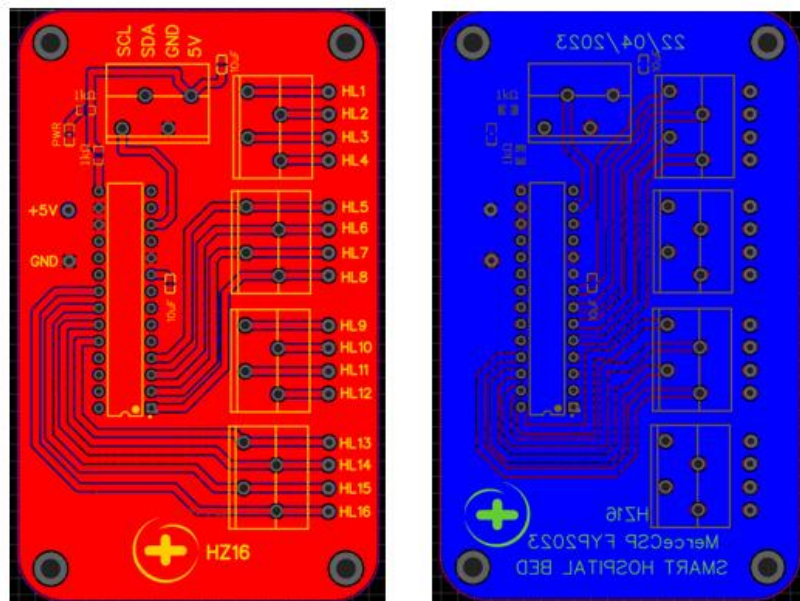


Figure 3.4.12: PCB for HZ16 Board: Front View (Left), Back View (Right)

### 3.5 Casing Design

As mentioned in earlier chapters, the main PCB board, touchscreen and speaker is placed into a casing that is mounted to the bed headboard. Figure 3.5.1 below showcases the front case of the 3D printed casing.

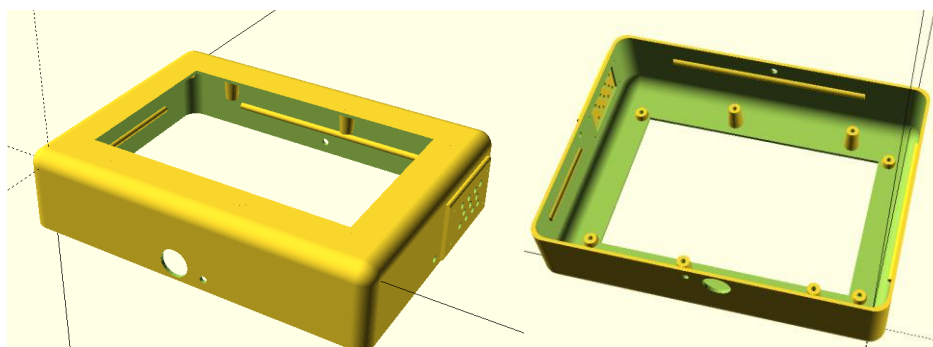


Figure 3.5.1: 3D-printed Front Case: Top View (Left), Bottom View (Right)

This casing is 148mm by 109 mm in size and has a front hole that has been designed to fit snugly with the Nextion 5-inch touchscreen. Some speaker holes have been added to the upper left of the front case for better sound clarity. The touchscreen and main board were secured to the casing via four screw mounts each. A discrete hole at the bottom meant for easy access for programming the microcontroller has also been added. This hole will be covered up when the back plate is attached, concealing the hole when it is not in use. Extrusions along the inner side of the front case help to act as placeholders for the back plate, allowing it to be flush with the front casing. Additionally, the back plate of this casing has a cable entry hole for wires to pass through without opening the casing.

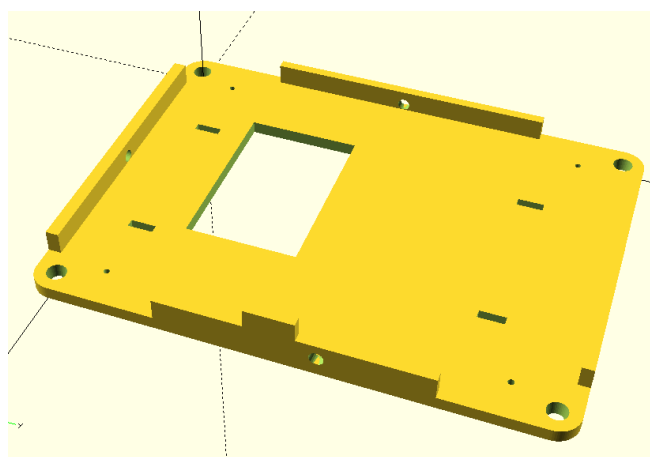


Figure 3.5.2: 3D-printed Back Plate



### 3.6 Software Development

For this instrumented bed monitoring system, the main microcontroller ESP32 was coded in C/C++ via the Arduino IDE. Moreover, the touchscreen interface was crafted with the Nextion HMI software, while HTML was utilised to program the web-based user interface.

#### 3.6.1 Sensor Array Reporting Process

The sensor array constructed consists of 128 nodes formed from a sixteen by eight matrix. As seen in Figure 3.6.1 below these nodes will be pulled high at rest. The VT8 board with equipped pull-up resistors acts as the input end, while HZ16 acts as the output, pulling down each of the sixteen sensor lines one after the other in a rolling manner. This is done by shifting bytes on patternA.

When the horizontal and vertical lines touch, the circuit becomes connected and continuous. The MCP23008 will read the input results as a single byte for each horizontal row, where for example: 10011111, indicating that there is pressure exerted at the second and third node from the left. This data is immediately stored and pushed to the screen, allowing for continuous monitoring.

```
int scan_bedArray() { //returns number of points under pressure

    uint8_t patternA = 0b11111110;
    uint8_t patternB = 0b11111110;
    uint8_t row;
    int yloc = 400; // Placement on LCD screen
    int totalNum = 0;
    int n, m;

    //Console.print("\n\n==== SCANNING BED =====");
    digitalWrite(LED_BUILTIN, HIGH);
    i2c_writeTwoByte(HZ16Addr, HZ16_GPIOA, 0b00000000, 0b00000000); // All pins preset to low but unconnected (INPUT)

    //Scan patient from feet to head
    for (int i = 0; i < 8; i++){ // Bottom half (8) of bed
        i2c_writeTwoByte(HZ16Addr, HZ16_DIRA, patternA, 0b11111111);
        delayMicroseconds(10); // delay a bit for line capacitance
        patternA = (patternA << 1) | 0b00000001;
        row = i2c_ReadByte(VT8Addr, VT8_GPIO);
        if (row == 0) { // if entire line zero suspect ghost, read again
            //beep(850,100);
            //Console.print("PortA Error");
            row = i2c_ReadByte(VT8Addr, VT8_GPIO);
        }
        n = draw_bedRow(row, yloc);
        bedArray[i] = row; // Store row
        totalNum = totalNum + n; // Sum no. of lit points
        yloc = yloc - 17; // 17 pixels vertical separation of dots on LCD
        delayMicroseconds(10); // delay to prevent reading reflections
    }
}
```

Figure 3.6.1: Partial code for scan\_bedArray()

### 3.6.2 Analysing Data for Triggering Alarms

Although the precise bed array information is stored, analyzing the exact positional information on the bed requires a lot of processing power and time. In order to simplify the process, the sum of bits, otherwise known as the total number of points in contact, is taken as our main variable of analysis. For one, the bed can be labelled vacant if there are less than five of contact, with consideration of stray contact points left behind by previous movement. Once the bed is deemed occupied, the alarm is triggered, and the countdown is shown onscreen.

```

if(occupiedFlag){
  showAlarmCountdown();
  check_alarmTimeout();
}
getRTCdatetime();
if (curr_ALARMmins == 0){
  LCD.print("t6.txt=\\" - Alarm Disabled -\\"\\x\\xFF\\xFF\\xFF");
}
totalPoints = scan_bedArray();
if (totalPoints > occpThreshold) { // At least e.g:5 points to be considered occupied
  if (!occupiedFlag){
    LCD.print("t0.txt=\\" OCCUPIED\\"\\x\\xFF\\xFF\\xFF");
    LCD.print("t0.bco=65504\\x\\xFF\\xFF\\xFF");
    Console.printf("\\nOccupied : Number of Points - %d", totalPoints);
    mute_flag = false;
    LCD.print("p3.pic=3\\x\\xFF\\xFF\\xFF");
    LCD.printf("play 0,%d,0\\x\\xFF\\xFF\\xFF", audio_occupied);
    occupiedFlag = true;
    seconds_left = curr_ALARMmins*60; // Restart Alarm with preset value
    storeStatusChg_time(1);
  }
  else{
    int chgPoints = totalPoints - oldtotalPoints;
    if(abs(chgPoints) > moveThreshold && oldtotalPoints > occpThreshold){
      LCD.print("t0.txt=\\" OCCUPIED\\"\\x\\xFF\\xFF\\xFF");
      LCD.print("t0.bco=65504\\x\\xFF\\xFF\\xFF");
      Console.printf("\\nPatient Moved : Number of Points - %d", totalPoints);
      seconds_left = curr_ALARMmins*60;
    }
  }
}
}

```

Figure 3.6.2: Partial Main Loop Code for Bed Movement

When the sum of bits remains constant for a prolonged period, it becomes an essential signal for healthcare professionals. This scenario may indicate that the patient has been stationary for an extended time, which can potentially lead to the development of bed sores. In such cases, the system can trigger an alarm to prompt timely repositioning, thus addressing a critical aspect of patient care. In Figure 3.6.2 above, this device is programmed to reset the alarm when the total number of contact points changed is above a certain threshold, effectively detecting major patient movement.

Conversely, if the sum of bits rapidly decreases, it serves as an effective indicator that the patient might be attempting to leave the hospital bed. The device recognizes this and proceeds to label the bed as vacant. A verbal alert is also played to alert healthcare providers promptly for assistance.

```

else{
  if (occupiedFlag){
    LCD.print("t0.txt=\"VACANT\"\\xFF\\xFF\\xFF");
    LCD.print("t0.bco=46522\\xFF\\xFF\\xFF");
    Console.print("\\nVacant");
    mute_flag = false;
    LCD.print("p3.pic=3\\xFF\\xFF\\xFF");
    LCD.printf("play 0,%d,0\\xFF\\xFF\\xFF", audio_vacant);
    occupiedFlag = false;
    LCD.print("t6.txt=\"----\"\\xFF\\xFF\\xFF");
    LCD.print("j0.val=100\\xFF\\xFF\\xFF");
    storeStatusChg_time(0);
  }
}
oldtotalPoints = totalPoints;

```

Figure 3.6.3: Partial Main Loop Code for Bed Occupancy

### 3.6.3 Touchscreen Communication

Following the instructions code given by Nextion, the ESP32 code was designed to receive all relevant serial messages to ensure smooth operation of the touchscreen GUI. As displayed in Figure 3.6.4 below, messages of differing nature would have different starting bytes.

```

void loop() {
  char ByteIN;
  size_t available = LCD.available();
  if (available > 0){
    ByteIN = LCD.read();
    if (ByteIN == 0x65) // touch event
      handleTouchEvent();
    else if (ByteIN == 0x88){
      Console.print("\\nLCD READY");
      flushLCD();
    }
    else if (ByteIN == 0x70) // incoming txt value
      handleIncomingString();
    else if (ByteIN == 0x71) // incoming num value
      handleIncomingNum();
    else {
      Console.print("\\nUnknown LCD Event: ");
      Console.print(ByteIN, HEX);
      flushLCD();
    }
  }
}

```

Figure 3.6.4: Partial Main Loop Code for Touchscreen Communication

Notably, touch events on the touchscreen will send the ESP32 messages with the string byte 0x65. The messages are then sorted by their page ID as seen in function `handleTouchEvent()`, before providing their specific item IDs where actions intended for each item is carried out for each case if touched.

```
void handleTouchEvent() {
    int pageNo;

    Console.print("\nTouched");
    delay(5); // delay to ensure other chars are coming in
    if (LCD.available()) {
        pageNo = LCD.read();
        Console.printf(" - Page:%d", pageNo);
        if (pageNo == page_main)
            handleMainPage();
        else if (pageNo == page_settings)
            handleSettingsPage();
        else if (pageNo == page_datetime)
            handleDateTimePage();
        else if (pageNo == page_network)
            handleNetworkPage();
        else
            Console.printf("\nUnknown Page:%d", pageNo);
    }
}
```

Figure 3.6.5: Code for `handleTouchEvent()`

### 3.6.4 Touchscreen GUI

As the Nextion Intelligent series touchscreen has been selected for this project, the Nextion HMI software has been employed to design an intuitive touchscreen interface. Figure 3.6.6 below shows the outline for the touchscreen main page. On the left, is a real time depiction of the sensor array, portraying the 128 points, where they are set to change color if pressure is detected.

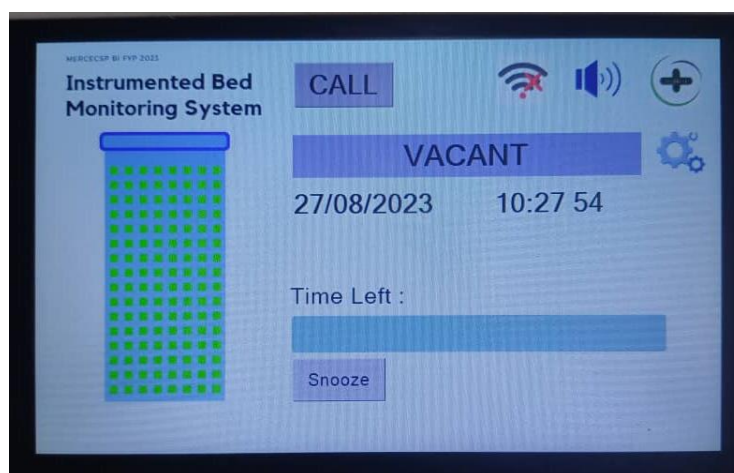


Figure 3.6.6: Touchscreen User Interface Main Page

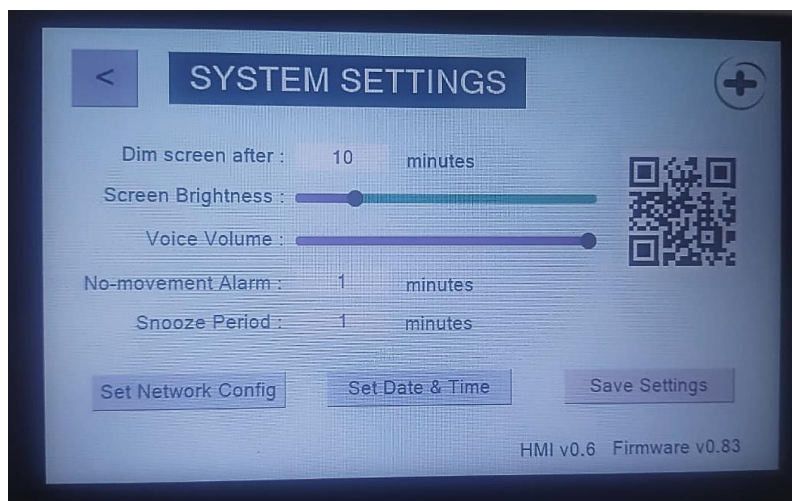


Figure 3.6.7: Touchscreen User Interface System Settings Page

The current bed status is displayed in full capitals on the right as ‘OCCUPIED’ or ‘VACANT’. When the bed is occupied, the alarm countdown is displayed, and the progress bar starts to move. The main page also has two buttons: a call button and a snooze button. A simple press on the settings icon will lead the user to the system settings page as in Figure 3.6.7.

Here, the screen brightness, sound volume, alarm period and snooze period can be set. A dynamic QR, linked to the web-based user interface, is also present on the right side of this page. From this settings page, the user can configure the network as well as set the date and time of the device. As a side note, however, the date and time is automatically synced to the internet time every midnight. Figures 3.6.8 and 3.6.9 below display the Network configuration page and Date & Time settings page.

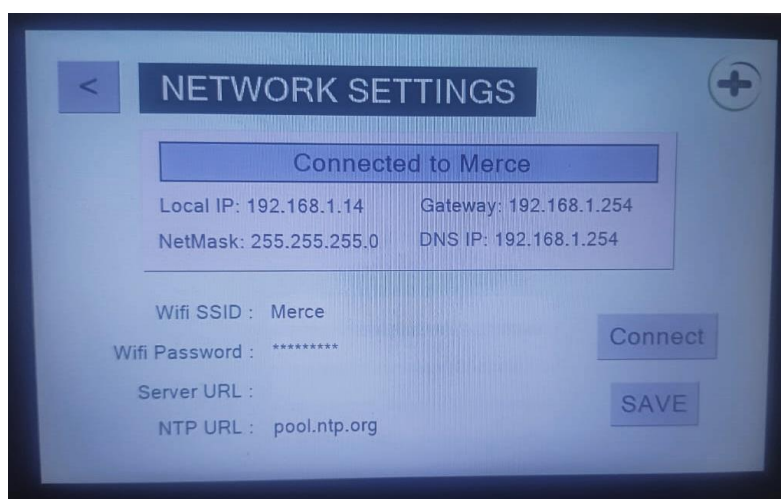


Figure 3.6.8: Touchscreen User Interface Network Settings Page

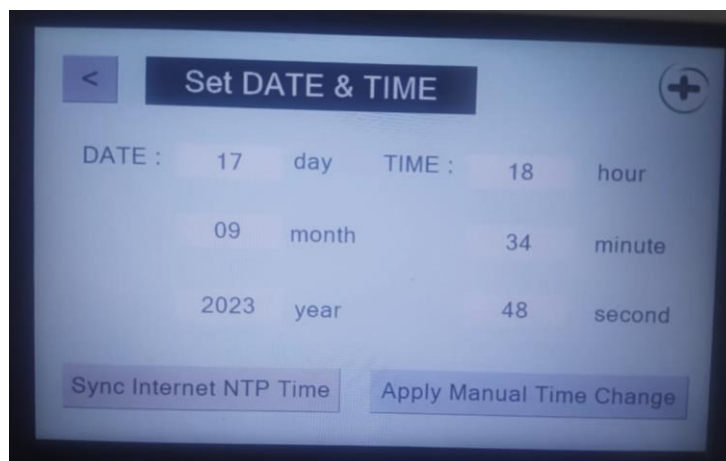


Figure 3.6.9: Touchscreen User Interface Date & Time Page

### 3.6.5 Web-based User Interface

In addition to the touchscreen interface, a web-based user interface has also been developed for this device. This allows for better accessibility and remote monitoring of the patient's bed status. The web-based user interface is accessible once the device is connected to the internet and contains similar elements to the touchscreen interface as seen in Figure 3.6.8 below. However, here the last five bed status history is consistently pulled to the screen, allowing nurses to know when this patient last left or returned to the hospital bed.

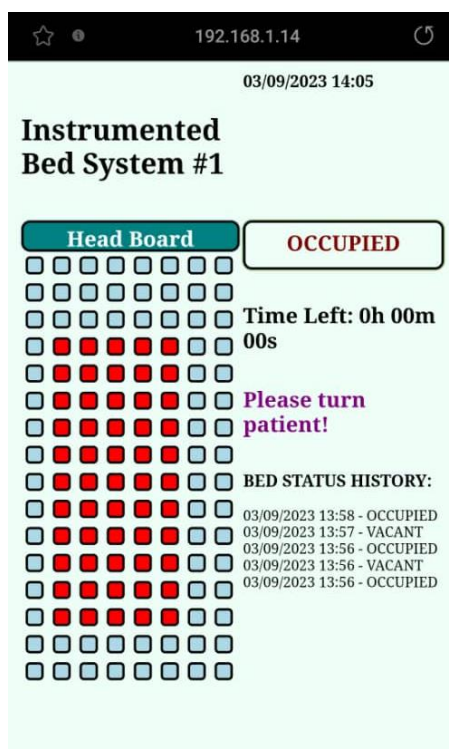


Figure 3.6.10: Web-based User Interface

### 3.7 Prototype Fabrication

The instrumented bed monitoring system prototype has been successfully constructed and is portrayed in the figures below. Its main components have been covered in the subsections prior and combined to form a working system.



Figure 3.7.1: Device console attached to headboard.



Figure 3.7.2: Assembled sensor array with stacked layers

Using the assembled sensor array consisting of two sensor layers and top bed sheet, the scanning of the array is done in real time, allowing for continuous monitoring of the patient's status. From the data collected from the sensor array, Time-To-Turn-Patient (TTTP) alarms and Bed-Exit (BE) alarms can be invoked, whereby TTTP alarms are designed to prevent pressure injuries by triggering when no bodily movement is detected after a fixed amount of time, while the BE alarms come into play when the patient enters or leaves the bed, categorising the bed status as occupied or vacant respectively.



Figure 3.7.3: Device user interface: Web-based (left), Touchscreen (Right)







It is observed that the sensor array is 99% accurate at detecting items with 350g in weight. In conclusion, the sensor array constructed can effectively detect items that are heavier than 350g in weight, which fulfils their intended function of detecting a human body as the average human hand weighs about 400 grams in total. Moreover, the detected output seems to be stable and does not provide flickering results with its once per second update rate.

#### **4.2.3 Sensor Array Working Current**

Since this system involves somewhat constant close contact with the body, the working current of the sensor array has also been investigated. With a 5V power source, the 12k pull-up resistors at each VT8 line causes the running current to be 0.4mA. When pressure is present and the two layers are in contact, this current will be shared. This means that the maximum current present at any single point in this sensor array is 3.3mA.

Considering the low current involved, there is no significant risk to the users in the case of a leakage current. Moreover, the stainless-steel wires were sewn into the fabric only via the bottom bobbin. This meant that the stainless-steel wires were only exposed to one side of the fabric, which effectively becomes the inner parts when the two sensor layers are stacked. Additionally, this sensor layer is meant to be used under the bed sheets which adds an extra layer of protection to the users.

#### **4.2.4 Stray Points on Sensor Array**

On the flip side, it was observed that the sensor array would not totally release upon the removal of pressure. Moreover, this phenomenon seemed to occur more around the edges where the cloth is strapped down. It is suspected that those near the edges of the bed had a higher tension and were more likely to not release when pressure is no longer applied. This was especially obvious when a large area of the sensor array is involved. A test was conducted to determine the average stray points left behind after a user gets up from laying down. Ten attempts of laying down and moving around on the bed for 5 minutes before getting up were made, and the results are as follows: 4,5,2,4,7,4,0,3,4,4.

From this, it was found that the average number of stray points left behind is 3.7 and that the pattern of stray points is entirely random, except for

its preference to occur near the bed edges. This number is within acceptable range as it is a smaller number than the average amount of contact points generated when one sits down on the edge of the bed. Henceforth, the threshold for bed status occupancy has been set to equal or larger than 5 total contact points. This threshold directly translates to the working efficiency of the system alerts which will be covered in later chapters. Additionally, it was found that these stray points were not stubborn in nature and easily reset their connection with slight movement exerted onto the bed, making the stray points change with every use.

#### 4.2.5 Sensor Array Reproduction Accuracy

Sensor array reproduction accuracy refers to the ability of a sensor array to faithfully reproduce the underlying physical phenomenon it is designed to detect, essentially measuring its overall precision and accuracy. In simpler terms, it measures how well the array can capture and represent the real-world data it is supposed to monitor.

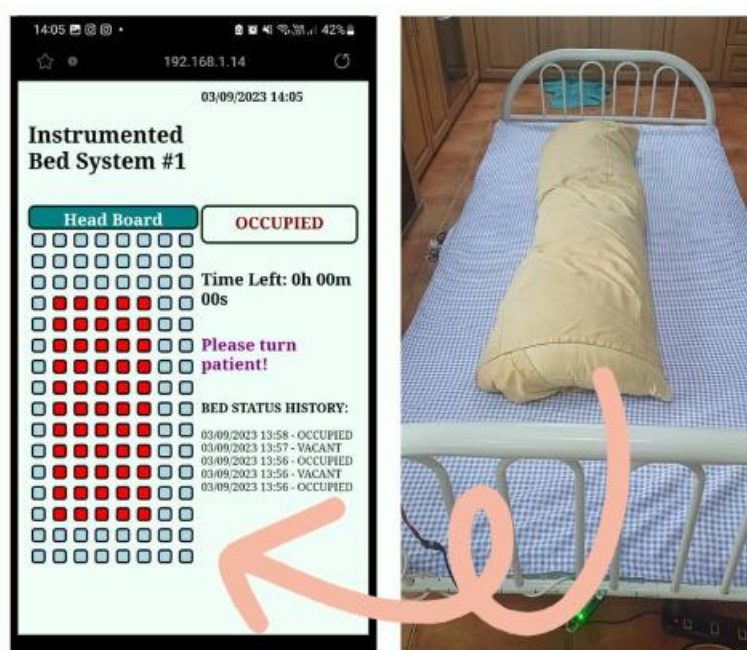


Figure 4.2.2: Example of Sensor Array Reproduction

For this test, a total of 6 laying positions were mapped and the data obtained was used for direct comparison. A spreadsheet has been used to record down the results of this test. Figure 4.2.2 below is one of the six positions tested,

where the highlighted sensor nodes indicate the true positive sensor nodes, those filled in with '4' are false positive results, while those blanked are true negatives.

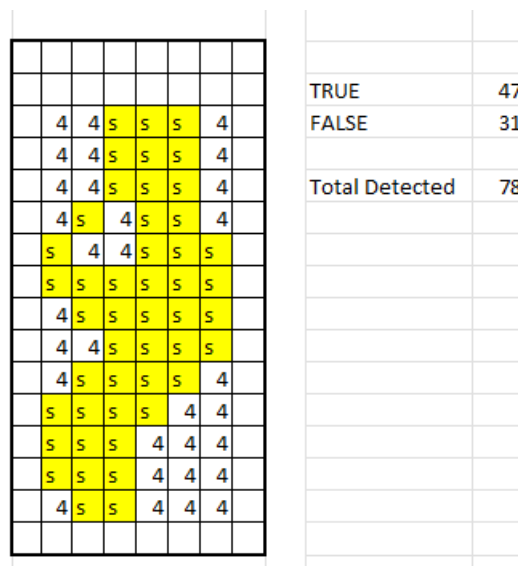


Figure 4.2.3: Sensor Array Reproduction of Side-laying Position

The average results of the six tests are summarised in the confusion matrix seen in table 4.2.3 below. As a side note, the false negatives were recorded in the confusion matrix, represents the stray points observed before and after laying down for this positional mapping. It can be seen that the results obtained is less than ideal, whereby the accuracy yielded for human shape reproduction is 70.3%. A key observation made was that the detection results have always come in a rectangular form or shape.

Table 4.2.3 Confusion Matrix for average human laying-down

Confusion matrix for laying-down		Actual Detection	
		Positive	Negative
Predicted Detection	Positive	49	33
	Negative	5	41

It is suspected that there is a key flaw in the initial sensor array design. For this device, only the testing of single node prototype to verify the proof of concept has been done prior to the full prototype. Before this full test, mostly singular small items were utilised to test the array functionality and this error was partially concealed. Since the observation made was that the detection

results have always come in a rectangular form or shape, it is suspected that there is a leakage current running from the actual contact point to other spots, causing false positives to appear significantly. In fact, it is suspected that the system is falsely detecting the digital low from other overlapping vertical points. Since the vertical line is continuous from the head to tail of bed, the digital low is believed to have travelled through it while the horizontal lines are being routinely pulled low, forming a 'Z' leakage pattern.

### **4.3 Efficacy of system alerts**

Moving forward, the efficacy of the actual system application, in the form of alarms and alerts, will be tested. While most of the results are linked to effects caused by the sensor array, the adequacy of the thresholds set shall be investigated.

#### **4.3.1 Time-to-turn-Patient Alarm**

The Time-to-turn-Patient (TTTP) alarm is an alarm designed to help aid in the mitigation of bed sores, alerting the healthcare providers when a patient requires repositioning. It can also function as a record of when the patient had last moved as the alarm is programmed to reset when movement is detected, whereby nurses coming for rounds can see whether the patient has recently moved on their own. If yes, the nurses can omit the repositioning, which both relieves workload from nurses and prevent the patients from being unnecessarily disturbed.

A total of fifty attempts of movement on the bed were made, with varying types from shuffling to simply turning from side to side. Figure 4.3.1 below is a pie chart of the successful alarm resets with movement detection.

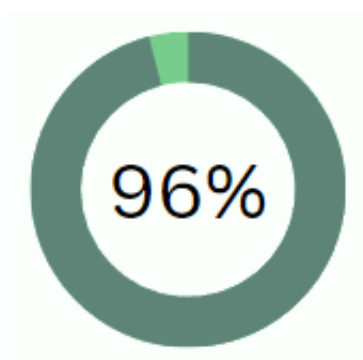


Figure 4.3.1: Pie Chart of Movement Detection for TTTP Alarm

Forty-eight out of fifty attempts were successful, yielding a 96% success rate in detecting movement. This proves that the movement threshold set in the system, whereby the change in total contact points being larger than 3, is quite appropriate for its intended function. This alarm accuracy can potentially be improved as other modes of analysis, other than the total number of contact points, is implemented for use in this device.

#### 4.3.2 Bed-Exit Alerts

The other main function of this device is the Bed-Exit (BE) alerts. Bed exit alerts are designed to enhance patient safety by providing timely notifications to healthcare providers when a patient attempts to leave their bed unassisted. As mentioned in the literature review, there are several bed exit monitors currently present in the market. However, most involve a small sensor pad spanning only the torso area and do not map the entire single bed.

For this device, the bed status changes from occupied to vacant and vice-versa following the threshold set with the total number of contact points, which has been set to equal or larger than 5 points of contact. Fifty attempts of getting on and getting off the bed each have been attempted and the results are shown in Figures 4.3.2 and 4.3.3 below.



Figure 4.3.2: Pie Chart of 'Getting ON' for Bed-Exit Alert

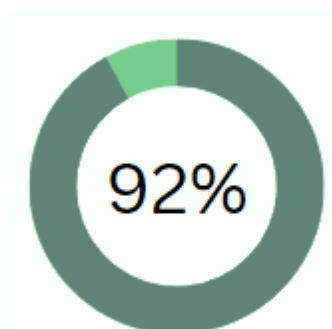


Figure 4.3.3: Pie Chart of 'Getting OFF' for Bed-Exit Alert

A full score of having 50 successful attempts has been recorded for getting onto the bed, changing the bed status from vacant to occupied. On the other hand, 46 out of fifty attempts getting off were successful, giving an alert accuracy of 92%. This meant that the system did not register that the patient had already left the bed, maintaining the occupied status despite it not being so on four separate occasions.

As discussed earlier, the occupancy threshold currently set for this device is 5 contact points. This was a value set based on the number of stray points left behind and is by theory appropriate for its function. During majority of the time, the device successfully works as intended. However, the results have shown that 8% of situations would involve more outliers than usual. Perhaps the earlier tests conducted for stray points were not sufficient and should be further investigated for a more comprehensive analysis of the sensor array behaviour.

#### **4.3.3 Rate of Scanning Bed Array**

At present, the system scans the bed array once per second and updates as so. Throughout the testing for various alarms and alerts, this scanning frequency has demonstrated its effectiveness, proving to be sufficient for its functions in continuous monitoring of the patient. There have been no notable incidents overlooked due to insufficient scanning speed. While it's possible to increase the frequency of bed array scans, the current rate represents a prudent compromise, setting aside processing power for potential additional functions to be executed by the ESP32.



## CHAPTER 5

### CONCLUSIONS AND RECOMMENDATIONS

#### 5.1 Conclusions

Although more improvements can be made to increase the sensor array accuracy, the current prototype has achieved its target objectives. At the moment, exact human shape reproduction is relatively low at 70.3%. However, current results are sufficient to detect human presence & movements, proven by the high accuracy of the alarms and alerts. Scanning of the array is done once per second in real time, allowing for continuous monitoring of the patient's status on the bed. The alerts and alarms are also activating as intended. Given that the prevention practices are dependent on nurse protocol, this device can aid in mitigating HACs, whereby the nurses are informed when a patient has roamed off the bed or has been immobile, enabling timely interventions for bed sores and falls.

#### 5.2 Recommendations for Future Work

While the current system has been designed to meet specific requirements, there is always room for enhancement to deliver more comprehensive and effective care. In fact, for this project, compromises have been made in order to accomplish the workload within the targeted timeline, while still striving to achieve the best result possible.

One critical aspect for improving patient monitoring systems is the enhancement of the sensor array. At the moment, there is a flaw in the key design that causes it to produce a rectangular output. Several ideas such as switching the outputs and inputs to cross check the data obtained can be considered. Additionally, the current sensor array records the data as a digital variable. To gather more detailed and nuanced data, consideration should be given to whether incorporating analogue data may be more suitable more this project.

Another recommendation is the implementation of more advanced processing of the current sensor data collected. The device currently operates

entirely on the total number of contact points in the sensor array. Other aspects of analysis are recommended in order to improve the device functionality. One such example is the analysis of the actual spatial positioning on the bed, identifying where exactly the pressure is being exerted.

Furthermore, if the sensor array accuracy is much improved, it might be suitable to implement a smart system that can identify the actual sleeping posture of the patient. Such AI could continuously analyse patient positions and, if necessary, recommend adjustments to alleviate pressure points. This proactive approach could help reduce the risk of pressure ulcers and improve patient comfort.

On top of that, it is recommended that a more comprehensive notification system is developed. At present, most of the alarms come in the form of sound and visual alerts, seen on the console and the web-based interface. While the web-based interface could potentially be developed to a dashboard for remote monitoring from for example a nurse station, an actual notification system to the caregiver's mobile or pager may be more effective.

## REFERENCES

Ahrq.gov. (2023). *Wandering Off the Floors: Safety and Security Risks of Patient Wandering*. [online] Available at: <https://psnet.ahrq.gov/web-mm/wandering-floors-safety-and-security-risks-patient-wandering> [Accessed 29 Apr. 2023].

Air Mattress Overlay (2022). *Stat Air™ Air Mattress Overlay*. [online] Blue Chip Medical. Available at: <https://www.bluechipmedical.com/mattress-overlays/air-water-overlays/stat-air/> [Accessed 30 Apr. 2023].

Amazon.com. (2023a). *Amazon.com: 45BSET-1Y Bed Alarm System by Secure Safety Solutions - 80 dB Alarm Monitor with Holder & 12" x 30" Bed Sensor Pad - Bed Alarms and Fall Prevention for Elderly Dementia Patient - Batteries Included : Industrial & Scientific*. [online] Available at: [https://www.amazon.com/Secure-45BSET-1Y-Alarm-Wandering-Prevention/dp/B01N7SX419?keywords=bed+alarms+for+elderly&qid=1673965870&sr=8-33&linkCode=ll1&tag=safewicom-20&linkId=ba0993402a65b0e61d42fd3dcadcd29d&language=en\\_US&ref\\_=as\\_li\\_ss\\_tl](https://www.amazon.com/Secure-45BSET-1Y-Alarm-Wandering-Prevention/dp/B01N7SX419?keywords=bed+alarms+for+elderly&qid=1673965870&sr=8-33&linkCode=ll1&tag=safewicom-20&linkId=ba0993402a65b0e61d42fd3dcadcd29d&language=en_US&ref_=as_li_ss_tl) [Accessed 29 Apr. 2023].

Amazon.com. (2023b). *Amazon.com: Lunderg Early Alert Bed Alarm System - Wireless Bed Sensor Pad & Pager - Elderly Monitoring Kit with Pre-Alert Smart Technology - Bed Alarms and Fall Prevention for Elderly and Dementia Patients : Health & Household*. [online] Available at: [https://www.amazon.com/Lunderg-Early-Alert-Alarm-System/dp/B08WJ7ZWQ7?keywords=bed+alarms+for+elderly&qid=1673965870&sr=8-5&linkCode=ll1&tag=safewicom-20&linkId=1e6bf7ed033632a88ba4bb082293e6e8&language=en\\_US&ref\\_=as\\_li\\_ss\\_tl](https://www.amazon.com/Lunderg-Early-Alert-Alarm-System/dp/B08WJ7ZWQ7?keywords=bed+alarms+for+elderly&qid=1673965870&sr=8-5&linkCode=ll1&tag=safewicom-20&linkId=1e6bf7ed033632a88ba4bb082293e6e8&language=en_US&ref_=as_li_ss_tl) [Accessed 29 Apr. 2023].

Anon, (2023). *Products - Data Briefs - Number 14 - February 2009*. [online] Available at: <https://www.cdc.gov/nchs/products/databriefs/db14.htm> [Accessed 30 Apr. 2023].

Azadeh Memarian, Somayeh Rauf Yazdinezhad, Shahrokh Mehrpisheh and Kamran Aghakhani (2015). Characteristics of absconders from a general health service, Rasoul Akram Hospital in 2013. *Polish annals of medicine*. [online] doi:<https://doi.org/10.1016/j.poamed.2015.04.011>.

Bansal, C., Scott, R., Stewart, D.J. and Cockerell, C.J. (2005). Decubitus ulcers: A review of the literature. *International Journal of Dermatology*, [online] 44(10), pp.805–810. doi:<https://doi.org/10.1111/j.1365-4632.2005.02636.x>.

Brecht Serraes, Martin van Leen, Jos M. G. A. Schols, Ann Van Hecke, Verhaeghe, S. and Beeckman, D. (2018). Prevention of pressure ulcers with a static air support surface: A systematic review. *International Wound Journal*, [online] 15(3), pp.333–343. doi:<https://doi.org/10.1111/iwj.12870>.

British Journal of Nursing. (2020). *Adult pressure area care: preventing pressure ulcers*. [online] Available at: <https://www.magonlinelibrary.com/doi/full/10.12968/bjon.2018.27.18.1050> [Accessed 30 Apr. 2023].

Callis, N. (2016). Falls prevention: Identification of predictive fall risk factors. *Applied Nursing Research*, [online] 29, pp.53–58. doi:<https://doi.org/10.1016/j.apnr.2015.05.007>.

Cheng (2015). Patient absconding behaviour in a public general hospital: retrospective study. *Hong Kong medical journal = Xianggang yi xue za zhi*, [online] 8(2). Available at: <https://pubmed.ncbi.nlm.nih.gov/11937662/> [Accessed 23 Apr. 2023].

Cheng, J., Sundholm, M., Zhou, B., Hirsch, M. and Lukowicz, P. (2016). Smart-surface: Large scale textile pressure sensors arrays for activity recognition. *Pervasive and Mobile Computing*, 30, pp.97–112. doi:<https://doi.org/10.1016/j.pmcj.2016.01.007>.

Eithne Keelaghan, Margolis, D.J., Zhan, M. and Baumgarten, M. (2008). Prevalence of pressure ulcers on hospital admission among nursing home residents transferred to the hospital. *Wound Repair and Regeneration*, [online] 16(3), pp.331–336. doi:<https://doi.org/10.1111/j.1524-475x.2008.00373.x>.

electron\_one (2020). *The role of capacitors in power supply and lighting circuits - OnElectronTech*. [online] OnElectronTech. Available at: <https://www.onelectrontech.com/role-of-capacitors-can-be-rc-filter-oscillator-time-constant-energy-storage-coupling-decoupling-bypass/#:~:text=1.,decoupling%2C%20filtering%20and%20energy%20storage.&text=Filtering%20is%20an%20important%20part,in%20almost%20all%20power%20circuits>. [Accessed 30 Apr. 2023].

Falkowski, J., Watts, V., Falkowski, W. and Dean, T. (1990). Patients Leaving Hospital Without the Knowledge or Permission of Staff–Absconding. *British Journal of Psychiatry*. [online] doi:<https://doi.org/10.1192/bjp.156.4.488>.

Fatt Soon Lee, Sondi Sararaks, Weng Keong Yau, Zen Yang Ang, Anis Syakira Jailani, Zulkarnain Abd Karim, Naing, L., Krishnan, T., Ai Reen Chu, Suria Junus, Mohd Sharifuddin Ahmad, Norhayaty Sapiee, Vicneas Wary Veloo, Manoharan, S. and Hamid, M. (2022). Fall determinants in hospitalised older patients: a nested case control design - incidence, extrinsic and intrinsic risk in Malaysia. *BMC Geriatrics*, [online] 22(1). doi:<https://doi.org/10.1186/s12877-022-02846-6>.

Gherzi, I., Mario Arturo González-Mariño and Mónica Teresita Miralles (2018). Smart medical beds in patient-care environments of the twenty-first century: a state-of-art survey. *BMC Medical Informatics and Decision Making*, [online] 18(1). doi:<https://doi.org/10.1186/s12911-018-0643-5>.

Hayn, D., Falgenhauer, M., Jürgen Morak, Wipfler, K., Willner, V., Liebhart, W. and Schreier, G. (2015). An eHealth System for Pressure Ulcer Risk Assessment Based on Accelerometer and Pressure Data. *Journal of Sensors*, [online] 2015, pp.1–8. doi:<https://doi.org/10.1155/2015/106537>.

Health, C. (2023). *Legacy 10, Single, Mattress Overlay inc Air Pump - CareWell Health*. [online] CareWell Health. Available at: <https://www.carewell.com.au/~19> [Accessed 30 Apr. 2023].

Hoque, E., Dickerson, R. and Stankovic, J. (n.d.). *Monitoring Body Positions and Movements During Sleep using WISPs*. [online] Available at: <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=40aa5cab015e5f2ba12fd6f9dac9cd0f0e1f6289> [Accessed 1 May 2023].

Jason Phil Seow, Tse Lert Chua, Fazila Aloweni, Lim, S. and Shin Yuh Ang (2022). Effectiveness of an integrated three-mode bed exit alarm system in reducing inpatient falls within an acute care setting. *Japan Journal of Nursing Science*. [online] doi:<https://doi.org/10.1111/jjns.12446>.

Kennedy, E. (2023). An Evidence-Based Approach to Protecting Our Biggest Organ: Implementation of a Skin, Surface, Keep Moving, Incontinence/Moisture, and Nutrition/Hydration (SSKIN) Care Bundle. *The*

*journal of doctoral nursing practice*, [online] 16(1), pp.62–80.  
doi:<https://doi.org/10.1891/jdnp-2021-0040>.

Kim, Y.-J. and Je Hyeok Oh (2020). Recent Progress in Pressure Sensors for Wearable Electronics: From Design to Applications. *DOAJ (DOAJ: Directory of Open Access Journals)*. [online] doi:<https://doi.org/10.3390/app10186403>.

Kogilavani Subermaniam, Ridgwan Welfred, Subramanian, P., Karuthan Chinna, Ibrahim, F., Mohktar, M.S. and Maw Pin Tan (2017). The Effectiveness of a Wireless Modular Bed Absence Sensor Device for Fall Prevention among Older Inpatients. *Frontiers in Public Health*, [online] 4.  
doi:<https://doi.org/10.3389/fpubh.2016.00292>.

Lastminuteengineers.com. (2023). Available at:  
<https://lastminuteengineers.com/esp32-pinout-reference/> [Accessed 25 Apr. 2023].

Laurino, M., Arcarisi, L., Carbonaro, N., Gemignani, A., Menicucci, D. and Tognetti, A. (2020). A Smart Bed for Non-Obtrusive Sleep Analysis in Real World Context. *IEEE Access*, [online] 8, pp.45664–45673.  
doi:<https://doi.org/10.1109/access.2020.2976194>.

LeLaurin, J.H. and Shorr, R.I. (2019). Preventing Falls in Hospitalized Patients. *Clinics in Geriatric Medicine*, [online] 35(2), pp.273–283.  
doi:<https://doi.org/10.1016/j.cger.2019.01.007>.

Lotfolah Afzali Borojeny, Albatineh, A.N., Ali Hasanpour Dehkordi and Reza Ghanei Gheshlagh (2020). The Incidence of Pressure Ulcers and its Associations in Different Wards of the Hospital: A Systematic Review and Meta-Analysis. *International Journal of Preventive Medicine*, [online] 11(10), p.171. doi:[https://doi.org/10.4103/ijpvm.ijpvm\\_182\\_19](https://doi.org/10.4103/ijpvm.ijpvm_182_19).

Mahnaz Moradpour, Mohammadreza Amiresmaili, Mahmood Nekoei-Moghadam and Dehesh, T. (2021). The reasons why patients abscond from public hospitals in southeastern Iran: a qualitative study. *Archives of public health*, [online] 79(1). doi:<https://doi.org/10.1186/s13690-021-00634-z>.

Malaysia, N. (2023). *Deluxe Bed Alarm System with Long Term Sensor Pad (20" x 30" pad)*. [online] NineLife - Malaysia. Available at:  
<https://www.ninelife.my/products/deluxe-bed-alarm-system-with-long-term-sensor-pad-20-x-30-pad?srsltid=AR57->

fAdQPAT86mfyij81znSjgAqBUUOO9hphsRb5KzVdIXvyJ6C5D148SU  
[Accessed 29 Apr. 2023].

McInnes, E., Asmara Jammali-Blasi, Sally, Dumville, J.C., Middleton, V. and Cullum, N. (2015). Support surfaces for pressure ulcer prevention. *The Cochrane library*, [online] 2015(9). doi:<https://doi.org/10.1002/14651858.cd001735.pub5>.

Medpage-ltd.com. (2020). (*TUMBKDRX*) *TumbleCare by Medpage Bed occupancy detection alarm system*. [online] Available at: [https://www.medpage-ltd.com/wandering-falls-bed-leaving-alarms?product\\_id=1727](https://www.medpage-ltd.com/wandering-falls-bed-leaving-alarms?product_id=1727) [Accessed 29 Apr. 2023].

Menéndez, M., José Antonio Alonso, J.C. Miñana, Arche, J.M., Díaz, J.S. and F. Vázquez (2013). Characteristics and associated factors in patient falls, and effectiveness of the lower height of beds for the prevention of bed falls in an acute geriatric hospital. *Revista De Calidad Asistencial*, [online] 28(5), pp.277–284. doi:<https://doi.org/10.1016/j.cali.2013.01.007>.

Moore, Z., Avsar, P., Conaty, L., Moore, D.H., O'Connor, T. and Patton, D. (2020). The prevalence of pressure ulcers in Europe, what does the European data tell us: a systematic review. *Journal of Wound Care*. [online] Available at: <https://www.magonlinelibrary.com/doi/abs/10.12968/jowc.2019.28.11.710> [Accessed 30 Apr. 2023].

Morse, J.M., Gervais, P., Pooler, C., Merryweather, A., Doig, A.K. and Bloswick, D.S. (2015). The Safety of Hospital Beds. *Global qualitative nursing research*, [online] 2, p.233339361557532-233339361557532. doi:<https://doi.org/10.1177/2333393615575321>.

Nice.org.uk. (2014). *1 Recommendations / Pressure ulcers: prevention and management / Guidance / NICE*. [online] Available at: <https://www.nice.org.uk/guidance/cg179/chapter/1-Recommendations#management-adults> [Accessed 30 Apr. 2023].

Npiap.com. (2019). *Pressure Injury Stages - National Pressure Ulcer Advisory Panel*. [online] Available at: <https://npiap.com/page/PressureInjuryStages> [Accessed 30 Apr. 2023].

Oliver, D. (2008). Falls risk-prediction tools for hospital inpatients. Time to put them to bed? *Age and Ageing*, [online] 37(3), pp.248–250. doi:<https://doi.org/10.1093/ageing/afn088>.

S3 ® MedSurg Bed. (n.d.). Available at: <https://www.stryker.com/content/dam/stryker/acute-care/products/s3/resources/S3BrochureMkt%20Lit223RevD7.pdf> [Accessed 30 Apr. 2023].

Singh, I., Okeke, J. and Edwards, C.J. (2015). Outcome of in-patient falls in hospitals with 100% single rooms and multi-bedded wards. *Age and Ageing*, [online] 44(6), pp.1032–1035. doi:<https://doi.org/10.1093/ageing/afv124>.

Syed and Sharma, S. (2022). *Pressure Ulcer*. [online] Nih.gov. Available at: [https://www.ncbi.nlm.nih.gov/books/NBK553107/#:~:text=Decubitus%20ulcers%2C%20also%20termed%20bedsores,than%20medial\)%2C%20and%20occur%20in%20the%20elderly](https://www.ncbi.nlm.nih.gov/books/NBK553107/#:~:text=Decubitus%20ulcers%2C%20also%20termed%20bedsores,than%20medial)%2C%20and%20occur%20in%20the%20elderly). [Accessed 30 Apr. 2023].

Toole, N., Meluskey, T. and Hall, N. (2016). A systematic review: barriers to hourly rounding. *Journal of Nursing Management*, [online] 24(3), pp.283–290. doi:<https://doi.org/10.1111/jonm.12332>.

Toye, C., Slatyer, S., Kitchen, S., Ingram, K., Bronson, M., Edwards, D., Welma van Schalkwyk, Pienaar, C., Wharton, P., Hill, K.D. and Hill, K.D. (2019). Bed Moves, Ward Environment, Staff Perspectives and Falls for Older People with High Falls Risk in an Acute Hospital: A Mixed Methods Study. *Clinical Interventions in Aging*, [online] Volume 14, pp.2223–2237. doi:<https://doi.org/10.2147/cia.s211424>.

Wilkie, T., Penney, S.R., Fernane, S. and Alexander (2014). Characteristics and motivations of absconders from forensic mental health services: a case-control study. *BMC Psychiatry*, [online] 14(1). doi:<https://doi.org/10.1186/1471-244x-14-91>.

Wood, V., Vindrola-Padros, C., Swart, N., McIntosh, M., Crowe, S., Morris, S. and Fulop, N. (2018). One to one specialising and sitters in acute care hospitals: A scoping review. *International Journal of Nursing Studies*, [online] 84, pp.61–77. doi:<https://doi.org/10.1016/j.ijnurstu.2018.04.018>.



World (2021). *Falls*. [online] Who.int. Available at: <https://www.who.int/news-room/fact-sheets/detail/falls> [Accessed 29 Apr. 2023].

World Health Organization (2008). *WHO Global Report on Falls Prevention in Older Age*. [online] World Health Organization. Available at: <https://www.who.int/publications/i/item/9789241563536> [Accessed 29 Apr. 2023].

Xsensor.com. (2021). *Pressure Injury 101: Stage 2 Pressure Ulcers*. [online] Available at: <https://blog.xsensor.com/stage-two-pressure-sores/> [Accessed 14 Sep. 2023].

Zhan, D. (2022). *NX8048P050-011C - Nextion*. [online] Nextion. Available at: <https://nextion.tech/datasheets/nx8048p050-011c/> [Accessed 25 Apr. 202

## APPENDICES

## Appendix A: Turnitin Report

## Bi\_1906112\_Turnitin

## ORIGINALITY REPORT

<b>1</b> %	<b>1</b> %	<b>0</b> %	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

## PRIMARY SOURCES

<b>1</b>	<b>pdffox.com</b> Internet Source	<1 %
<b>2</b>	<b>pcbdesignworld.com</b> Internet Source	<1 %
<b>3</b>	<b>Eithne Keelaghan. "Prevalence of pressure ulcers on hospital admission among nursing home residents transferred to the hospital", Wound Repair and Regeneration, 5/2008</b> Publication	<1 %
<b>4</b>	<b>www.hqsc.govt.nz</b> Internet Source	<1 %
<b>5</b>	<b>eprints.utar.edu.my</b> Internet Source	<1 %
<b>6</b>	<b>www.tetsa.ir</b> Internet Source	<1 %
<b>7</b>	<b>careersdocbox.com</b> Internet Source	<1 %
<b>8</b>	<b>kth.diva-portal.org</b> Internet Source	<1 %

## Appendix B: Microcontroller Code

## /\* Version History LOG

04/05/2023 V0.1 able to talk to 5 inch screen - startingup  
V0.2 using UART2 for LCD + Buzzer (beep) working

06/05/2023 V0.3 able to detect touch event from LCD  
V0.4 able to detect page number and item ID from LCD  
V0.5 LCD Baud rate changed to 115200  
Mute/Unmute Icon Working  
V0.6 Able to play sound files (through speaker) on touch  
V0.7 Play voice when Mute/unmute

07/05/2023 V0.8 Serial console using 115200 Baud  
Settings Page workable  
V0.9 Multi-page handling from LCD  
Can read and store Wifi SSID and password from LCD  
V0.10 Able to read RTC Date from LCD  
V0.11 Plays Jingle at Startup  
Push Date and Time from RTC to LCD at Startup  
V0.12 Show Time left and countdown progress bar  
Call Nurse and Snooze button added  
V0.13 Snooze button to extend alarm time by 1minute for each press  
Can select Alarm period in settings page

08/05/2023 V0.14 Setup Set date and time page  
snoozing can only be done up to half the set alarm period  
V0.15 Can input and receive date and time set by user  
V0.16 User can set date and time on RTC  
v0.17 Current date and time values preset on set date time page upon opening page

19/05/2023 v0.18 Added Date & time validation before push to screen

20/05/2023 v0.19 Moved Network settings to New page  
Using HMI Ver 0.4  
Can collect server and NTP url  
v0.20 Collect and store the setting variables with save button

21/05/2023	v0.21 Can adjust brightness and volume in real time Can set No-movement alarm period and snooze period
	v0.22 Can disable alarm by setting alarm period to 0 minutes
25/05/2023	v0.23 Can Store brightness & volume persistently using EEPROM V0.24 Can store dimtime, alarmtime & snooze persistently using EEPROM
27/05/2023	V0.25 Can power on & off peripheral boards
28/05/2023	V0.27 Able to detect VT8 and HZ16 boards at startup V0.29 Using multiple .ino files V0.31 Able to read VT8 GPIO values !
29/05/2023	V0.32 Able to write to HZ16 DIR & GPIO pins as twinbytes V0.33 Scan Sweep of HZ16 done V0.34 Able to read bed sensing values Sped up reading of VT8 by using microseconds delay. V0.36 Able to draw bed array image from sensing array on lcd
30/05/2023	V0.38 Initialise VT8 INPUT polarity register at startup & hardreset Changed to HMI V0.5 - Changed background format V0.40 Clears Mute when bed status changes
15/06/2023	V0.41 Can connect to Wifi V0.42 Save Wifi SSID & Password persistently in EEPROM V0.43 Use saved details to connect directly to wifi
16/06/2023	V0.44 Can detect Wifi status up or down V0.45 able to connect to wifi at startup & disable wifi V0.46 can set host name for esp32 Wifi V0.47 collect netMask, gatewayIP and DNS Able to detect problems with Input register and IOCON and auto reset within 30 secs V0.48 Using HMI Ver 0.6 - can show wifi logo & firmware Ver in settings page Can display Network status & info on Network page

- 17/06/2023 V0.49 Corrected bugs with displaying network info  
V0.50 Corrected entire row detected bug  
V0.51 Auto detect boards (VT8 & HZ16) error every 10 secs and self-recover  
V0.52 Added NTPClient to sync time to RTC every in setdatetime page & every 12 hours
- 18/06/2023 V0.53 Can detect NTP sync failure  
V0.55 Can save NTP Server URL persistently in EEPROM  
Will use default NTP server if NTP URL not set  
V0.56 Changes to NTP URL takes effect immediately, NTP server initialised upon Wifi-connection  
V0.57 Changed ShiftOut code to resolve false positives  
V0.58 Added Dynamic QR code leading to internal IP web page when connected to wifi  
V0.59 HTTP Server code started (hello world)  
V0.60 HTTP server verified to work with android phone & pc chrome
- 21/06/2023 V0.61 Changes to HTTP code to prevent favicon request + other html details
- 25/06/2023 V0.62 HTML Table added  
V0.64 Webpage to show patient calling nurse
- 26/06/2023 V0.65 Webpage showing bed occupied/vacant
- 02/07/2023 V0.66 Added delay to after each pulldown on bed scan array to avoid human capacitance problem
- 09/07/2023 V0.67 Webpage adjustments  
V0.68 More webpage adjustments  
V0.70 Added bed array boxes to webpage  
Webpage can see alarm timer and turn patient alert
- 06/08/2023 V0.72 Change bed occupied status to more than 4 points  
Removed beep for entire line ghost  
V0.73 Add delays between line readings to handle reflections  
V0.74 HTML page reflects bed array in real time
- 20/08/2023 V0.78 Added Button to clear Call Nurse Alert

```

                V0.79  Moved HTTP items to separate functions
21/08/2023      V0.80  Auto refresh webpage every 10 secs
*/
#include <Arduino.h>
#include <EEPROM.h>
#include <WiFi.h>
#include <time.h>

#define version 83
#define bedNo 1

#define Console Serial
#define LCD Serial1
#define LCDBAUD 115200    // 115200

#define occpThreshold 5    // Occupied Number of Sensor points threshold
#define moveThreshold 3

#define VT8Addr 0x40    // 40hex when taking 8 bits with A0, A1 and A2
= (0,0,0) // 20h for 7bit
#define HZ16Addr 0x42    // 42hex when taking 8 bits with A0, A1 and A2
= (1,0,0)
#define VT8_IPOL 0x01    // Input Polarity register
#define VT8_GPIO 0x09
#define HZ16_GPIOA 0x12
#define HZ16_GPIOB 0x13
#define HZ16_DIRA 0x00
#define HZ16_DIRB 0x01
#define HZ16_IOCON1 0x0A
#define HZ16_IOCON2 0x05

// Pin Definition
#define LED_BUILTIN 2

```

```
#define BUZZPIN 13
#define LCD_RXPIN 16 // GPIO 16 => RX for LCD
#define LCD_TXPIN 17 // GPIO 17 => TX for LCD - UART2
#define IOPW_PIN 15 // GPIO 15 -> Power Secondary boards
#define SDA_PIN 21
#define SCL_PIN 22

//page
#define page_main 0
#define page_settings 1
#define page_datetime 3
#define page_network 5

//main page components
#define id_settings 4
#define id_speaker 10
#define id_call 12
#define id_snooze 15

//settings page components
#define id_settingsback 3
#define id_dimtime 19
#define id_brightness 7
#define id_voicevol 8
#define id_alarmperiod 20
#define id_setsnooze 21
#define id_setdatetime 6
#define id_setnetwork 13
#define id_settingssave 16

// network page components
#define id_networkback 7
#define id_ssid 5
```

```
#define id_pass 6
#define id_server 10
#define id_ntp 12
#define id_connect 9
#define id_networksave 13

//date time page components
#define id_datetimeback 2
#define id_changedatetime 18
#define id_syncntp 19

#define id_YEAR 9
#define id_MONTH 12
#define id_DAY 11
#define id_HOUR 13
#define id_MINUTE 14
#define id_SECOND 10

// audio list
#define audio_occupied 0
#define audio_vacant 1
#define audio_mute 2
#define audio_soundON 3
#define audio_sysReady 4
#define audio_bedSysReady 5
#define audio_callNurse 6
#define audio_turnPatient 7
#define audio_testSpeech 8

// Wifi Element Flag
#define get_SSID 51
#define get_PASS 52
#define get_ServerURL 53
```



```
#define get_NTPURL 54

// RTC Time Flag
#define get_RTCYEAR 1
#define get_RTCMONTH 2
#define get_RTCDAY 3
#define get_RTCHOUR 4
#define get_RTCMINUTE 5
#define get_RTCSECOND 6

// Set Time Flag
#define get_SETYEAR 7
#define get_SETMONTH 8
#define get_SETDAY 9
#define get_SETHOUR 10
#define get_SETMINUTE 11
#define get_SETSECOND 12

// Settings Elements
#define get_DIMTIME 21
#define get_BRIGHTNESS 22
#define get_BRIGHTNESSNOW 23
#define get_VOICEVOL 24
#define get_VOICEVOLNOW 25
#define get_ALARM 26
#define get_SNOOZE 27

// Screen-page Timeout element
#define timeout_max 60
#define timeout_extension 30

//EEPROM storage locations
#define EEaddr_bright 0
```

```
#define EEaddr_vol 1
#define EEaddr_dimtime 2
#define EEaddr_alarm 3
#define EEaddr_snooze 4
#define EEaddr_ssid 10
#define EEaddr_pass 44
#define EEaddr_ntp 78
//#define EEaddr_server 102

//Wifi status
#define wifi_idle 0
#define wifi_connecting 1
#define wifi_connected 2
#define wifi_down 3

bool mute_flag = false;
bool VT8_existFlag = false;
bool HZ16_existFlag = false;
bool occupiedFlag = false;
bool BedErrFlag = false;
bool callNurseFlag = false;

int totalPoints; // Number of Points pressed on bed array
int oldtotalPoints;

int get_flag = 0;
int seconds_left; // 2 hours is 7200sec** // Time left before alarm
int secs;
int oldsecs;
int rollingMins = 0;
int rollingHour = 0;
int page_timeout = 0;
```

```
int curr_YEAR = 2023;
int curr_MONTH = 5;
int curr_DAY = 7;
int curr_HOUR = 14;
int curr_MINUTE = 45;
int curr_SECOND = 0;

int curr_DIMMINS;
int curr_BRIGHTNESS;
int curr_VOLUME;
int curr_ALARMMINS;
int curr_SNOOZE;

int setDay;
int setMonth;
int setYear;
int setHour;
int setMinute;
int setSecond;

int timerHour;
int timerMins;
int timerSecs;

int wifiStatus = 0;

int statusTime_1[10]; // to save and display the previous time of occupied
and vacant
int statusTime_2[10];
int statusTime_3[10];
int statusTime_4[10];
int statusTime_5[10];
```

```
char Wifi_SSID[33];
char Wifi_PASS[33];
char Wifi_ServerURL[33];
char Wifi_NTPURL[33];

uint8_t bedArray[16];    // Temporary Storage for Bed Array

const char* HostName = "MerceFYP-BED";
const char* defaultNTPserver = "pool.ntp.org";
const long  gmtOffset_sec = 28800 * 1;
const int   daylightOffset_sec = 0 * 0;

WiFiServer HTTPserver(80);

void setup() {

  pinMode(BUZZPIN, OUTPUT);
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(IOPW_PIN, OUTPUT);

  Console.begin(115200);
  LCD.begin(LCDBAUD, SERIAL_8N1, LCD_RXPIN, LCD_TXPIN);
  powerON_bedArray();

  beep(800,200); // Play Jingle to indicate restart
  delay(200);
  beep(900,200);
  delay(200);
  beep(1100,600);
  delay(400);

  Console.println("\nStarting Instrumented Bed Monitoring System");
```

```
LCD.printf("page %d\\xFF\\xFF\\xFF", page_main); // Make sure LCD is st
MAIN Page at startup
```

```
LCD.print("t0.txt=\\\"STARTING UP\\\"\\xFF\\xFF\\xFF");
```

```
LCD.print("t4.txt=\\\" \\\"\\xFF\\xFF\\xFF\"); // clear t4 (Error msg line)
```

```
int IOboards_ok = IOboards_Check();
```

```
if (HZ16_existFlag)
```

```
    init_HZ16();
```

```
delay(200);
```

```
if (!IOboards_ok){
```

```
    LCD.print("t0.bco=63488\\xFF\\xFF\\xFF"); // background colour to
```

```
RED
```

```
    LCD.print("t0.txt=\\\"BED ARRAY ERROR\\\"\\xFF\\xFF\\xFF");
```

```
    beep(700,600);
```

```
}
```

```
else {
```

```
    i2c_WriteByte(VT8Addr, VT8_IPOL, 0b00000000); // Make sure input
polarity register is not inverted
```

```
    i2c_WriteByte(HZ16Addr, HZ16_IOCON1, 0b00000000); // Make sure
IOCON register is all zeroised
```

```
    i2c_WriteByte(HZ16Addr, HZ16_IOCON2, 0b00000000); // Make sure
IOCON register is all zeroised
```

```
LCD.print("t0.bco=32725\\xFF\\xFF\\xFF");
```

```
LCD.print("t0.txt=\\\"READY\\\"\\xFF\\xFF\\xFF"); // Change BED STATUS
```

```
LCD.printf("play 0,%d,0\\xFF\\xFF\\xFF", audio_bedSysReady);
```

```
}
```

```
EEPROM.begin(200); // Initialize 200 bytes of EEPROM
```

```
curr_BRIGHTNESS = EEPROM.read(EEaddr_bright);
```

```
curr_VOLUME = EEPROM.read(EEaddr_vol);
```

```
curr_DIMMINS = EEPROM.read(EEaddr_dimtime);
```

```

curr_ALARMMINS = EEPROM.read(EEaddr_alarm);
curr_SNOOZE = EEPROM.read(EEaddr_snooze);
readSSIDFromEEPROM();
readPASSFromEEPROM();
readNTPURLFromEEPROM();

Console.printf("\nEEPROM Wifi SSID: %s", Wifi_SSID);
Console.printf("\nEEPROM Wifi Password: %s", Wifi_PASS);
Console.printf("\nEEPROM NTP URL: %s", Wifi_NTPURL);

seconds_left = curr_ALARMMINS*60;

statusTime_1[0] = 0;    // to save and display the previous time of occupied
and vacant
statusTime_2[0] = 0;
statusTime_3[0] = 0;
statusTime_4[0] = 0;
statusTime_5[0] = 0;

}

void loop() {

char ByteIN;

size_t available = LCD.available();
if (available > 0){
  ByteIN = LCD.read();
  if (ByteIN == 0x65) // touch event
    handleTouchEvent();
  else if (ByteIN == 0x88){
    Console.print("\nLCD READY");
    flushLCD();
  }
}
}

```

```

}
else if (ByteIN == 0x70) // incoming txt value
    handleIncomingString();
else if (ByteIN == 0x71) // incoming num value
    handleIncomingNum();
else {
    Console.print("\nUnknown LCD Event: ");
    Console.print(ByteIN, HEX);
    flushLCD();
}
}
CheckHTTPRequests();
secs = millis()/1000;
if (secs != oldsecs) { // Every Second
    oldsecs = secs;
    if(page_timeout) { //Not in Main Page
        page_timeout--;
        if(!page_timeout){ // Timeout for non-MainPage
            beep(850,50);
            showMainPage(); //back to main page
        }
    }
}
else { //In Main Page
    if(secs%3 == 0) { //Every three seconds
        if(wifiStatus == wifi_connecting || wifiStatus == wifi_down)
            checkwifiConnected();
        else if(wifiStatus == wifi_connected) {
            checkwifiDisconnected();
        }
    }
}
if(secs%10 == 0) { // every 10 secs
    checkBedStatus();
    bool status = IOboards_Check();
}

```

```

if (status && BedErrFlag){
    LCD.print("t0.bco=32725\xFF\xFF\xFF");
    LCD.print("t0.txt=\"READY\"\xFF\xFF\xFF"); // Change BED
STATUS
    LCD.printf("play 0,%d,0\xFF\xFF\xFF", audio_bedSysReady);
    BedErrFlag = false;
}
}
if(secs%60 == 0) { // every minute
    rollingMins++;
    if(wifiStatus == wifi_idle){
        if(strlen(Wifi_SSID) > 1 ){
            connectWIFI();
            beep(1000,200);
        }
    }
    if (rollingMins%60 == 0) { // once every hour
        rollingHour++;
        if(wifiStatus == wifi_connected) {
            syncNTPtime();
            beep(700,100);
        }
    }
}

if(occupiedFlag){
    showAlarmCountdown();
    check_alarmTimeout();
}
getRTCdatetime();
if (curr_ALARMMINS == 0){
    LCD.print("t6.txt=\"- Alarm Disabled -\"\xFF\xFF\xFF");
}

```



```

totalPoints = scan_bedArray();
if (totalPoints > occpThreshold) {           // At least e.g:5 points to be
considered occupied
    if (!occupiedFlag){
        LCD.print("t0.txt=\"OCCUPIED\"\\xFF\\xFF\\xFF");
        LCD.print("t0.bco=65504\\xFF\\xFF\\xFF");
        Console.printf("\\nOccupied : Number of Points - %d", totalPoints);
        mute_flag = false;
        LCD.print("p3.pic=3\\xFF\\xFF\\xFF");
        LCD.printf("play 0,%d,0\\xFF\\xFF\\xFF", audio_occupied);
        occupiedFlag = true;
        seconds_left = curr_ALARMMINS*60;    // Restart Alarm with preset
value
        storeStatusChg_time(1);
    }
    else{
        int chgPoints = totalPoints - oldtotalPoints;
        if(abs(chgPoints) > moveThreshold && oldtotalPoints >
occpThreshold){
            LCD.print("t0.txt=\"OCCUPIED\"\\xFF\\xFF\\xFF");
            LCD.print("t0.bco=65504\\xFF\\xFF\\xFF");
            Console.printf("\\nPatient Moved : Number of Points - %d", totalPoints);
            seconds_left = curr_ALARMMINS*60;
        }
    }
}
else{
    if (occupiedFlag){
        LCD.print("t0.txt=\"VACANT\"\\xFF\\xFF\\xFF");
        LCD.print("t0.bco=46522\\xFF\\xFF\\xFF");
        Console.print("\\nVacant");
        mute_flag = false;
        LCD.print("p3.pic=3\\xFF\\xFF\\xFF");
    }
}
}

```

```
LCD.printf("play 0,%d,0\xFF\xFF\xFF", audio_vacant);
occupiedFlag = false;
LCD.print("t6.txt=\"----\"\xFF\xFF\xFF");
LCD.print("j0.val=100\xFF\xFF\xFF");
storeStatusChg_time(0);
}
}
oldtotalPoints = totalPoints;
}
}
}
```

```
void handleTouchEvent() {

int pageNo;

Console.print("\nTouched");
delay(5); // delay to ensure other chars are coming in
if (LCD.available()) {
    pageNo = LCD.read();
    Console.printf(" - Page:%d", pageNo);
    if (pageNo == page_main)
        handleMainPage();
    else if (pageNo == page_settings)
        handleSettingsPage();
    else if (pageNo == page_datetime)
        handleDateTimePage();
    else if (pageNo == page_network)
        handleNetworkPage();
    else
        Console.printf("\nUnknown Page:%d", pageNo);
}
}
```

```

void handleMainPage(){

int ItemID;

if (LCD.available()) {
    ItemID = LCD.read();
    Console.printf(" Item ID:%d", ItemID);
    flushLCD();
    beep(1200,50);
    if(ItemID == id_speaker){
        if(mute_flag){
            mute_flag = false;
            LCD.print("p3.pic=3\xFF\xFF\xFF");
            Console.printf(" MUTE OFF");
            LCD.printf("play 0,%d,0\xFF\xFF\xFF", audio_soundON);
        }
        else{
            mute_flag = true;
            LCD.print("p3.pic=4\xFF\xFF\xFF");
            Console.printf(" MUTE ON");
            LCD.printf("play 0,%d,0\xFF\xFF\xFF", audio_mute);

        }
    }
}
else if(ItemID == id_settings){
    Console.print(" Settings Pressed");
    page_timeout = timeout_max;
    LCD.printf("page %d\xFF\xFF\xFF", page_settings); // Go to settings Page
    LCD.printf("h0.val=%d\xFF\xFF\xFF",curr_BRIGHTNESS);
    LCD.printf("h1.val=%d\xFF\xFF\xFF",curr_VOLUME);
    LCD.printf("n0.val=%d\xFF\xFF\xFF",curr_DIMMINS);
    LCD.printf("n1.val=%d\xFF\xFF\xFF",curr_ALARMMINS);
}
}
}

```

```

LCD.printf("n2.val=%d\xFF\xFF\xFF",curr_SNOOZE);
LCD.printf("t8.txt=\"Firmware v0.%d\"\xFF\xFF\xFF", version);
if (wifiStatus == wifi_connected){
    const char* localIP = WiFi.localIP().toString().c_str();
    if (strlen(localIP) == 0){
        String strIP = IPtoString(WiFi.localIP());
        LCD.printf("qr0.txt=\"http://%s\"\xFF\xFF\xFF", strIP);
    }
    else
        LCD.printf("qr0.txt=\"http://%s\"\xFF\xFF\xFF", localIP);
}
else
    LCD.print("qr0.txt=\"https://www.utar.edu.my/\xFF\xFF\xFF");
}
else if(ItemID == id_call){
    Console.print(" CALL button Pressed");
    LCD.printf("play 0,%d,0\xFF\xFF\xFF",audio_callNurse);

    callNurseFlag = true;

    //i2c_WriteByte(HZ16Addr, HZ16_IOCON1, 0b11111111); // test to mess
up IOCON

    //i2c_WriteByte(VT8Addr, VT8_IPOL, 0b00001111); // test to mess up
Input polarity
}
else if(ItemID == id_snooze){
    Console.print(" SNOOZE button Pressed");
    if(seconds_left < ((curr_ALARMMINS*60)/2)) // stop excessive snoozing
        seconds_left = seconds_left + curr_SNOOZE*60;
    if (occupiedFlag){
        LCD.print("t0.txt=\"OCCUPIED\"\xFF\xFF\xFF");
        LCD.print("t0.bco=65504\xFF\xFF\xFF");
    }
}

```

```

    }
}
else {
    Console.printf(" Unknown Main Page ID:%d",ItemID);
}
}
}

void handleSettingsPage(){

int ItemID;

if (LCD.available()) {
    ItemID = LCD.read();
    Console.printf(" Item ID:%d", ItemID);
    flushLCD();
    beep(1200,50);
    if(ItemID == id_settingsback)
        showMainPage(); //back to main page
    else if (ItemID == id_settingsave){
        Console.printf(" Settings SAVE button Pressed - ");
        get_flag = get_DIMTIME;
        LCD.printf("get n0.val\xFF\xFF\xFF");
        //showMainPage();
    }
    else if(ItemID == id_setdatetime){
        page_timeout = timeout_max;
        LCD.printf("page %d\xFF\xFF\xFF", page_datetime); // Go to set date time
page
        delay(1);
        LCD.printf("n2.val=%d\xFF\xFF\xFF", curr_DAY); // initialise page with
current date and time values
        LCD.printf("n3.val=%d\xFF\xFF\xFF", curr_MONTH);

```

```

LCD.printf("n0.val=%d\xFF\xFF\xFF", curr_YEAR);
LCD.printf("n4.val=%d\xFF\xFF\xFF", curr_HOUR);
LCD.printf("n5.val=%d\xFF\xFF\xFF", curr_MINUTE);
LCD.printf("n1.val=%d\xFF\xFF\xFF", curr_SECOND);
}
else if(ItemID == id_setnetwork){
    page_timeout = timeout_max;
    LCD.printf("page %d\xFF\xFF\xFF", page_network); // Go to set network
page
    delay(1);
    LCD.printf("t3.txt=\"%s\"\xFF\xFF\xFF",Wifi_SSID);        // set current
values to page
    LCD.printf("t4.txt=\"%s\"\xFF\xFF\xFF",Wifi_PASS);
    LCD.printf("t8.txt=\"%s\"\xFF\xFF\xFF", Wifi_NTPURL);
    if(wifiStatus == wifi_connected){
        LCD.printf("t9.txt=\"Connected to %s\"\xFF\xFF\xFF", Wifi_SSID);
        LCD.print("t9.bco=46522\xFF\xFF\xFF");
        const char* localIP = WiFi.localIP().toString().c_str();
        if (strlen(localIP) == 0){
            String strIP = IPtoString(WiFi.localIP());
            LCD.printf("t11.txt=\"Local IP: %s\"\xFF\xFF\xFF", strIP);
        }
        else
            LCD.printf("t11.txt=\"Local IP: %s\"\xFF\xFF\xFF", localIP);

        const char* netMask = WiFi.subnetMask().toString().c_str();
        if (strlen(netMask) == 0){
            String strMask = IPtoString(WiFi.subnetMask());
            LCD.printf("t12.txt=\"NetMask: %s\"\xFF\xFF\xFF", strMask);
        }
        else
            LCD.printf("t12.txt=\"NetMask: %s\"\xFF\xFF\xFF", netMask);

```

```

const char* Gateway = WiFi.gatewayIP().toString().c_str();
if (strlen(Gateway) == 0){
    String strGate = IPtoString(WiFi.gatewayIP());
    LCD.printf("t13.txt=\\"Gateway: %s\\"\\xFF\\xFF\\xFF", strGate);
}
else
    LCD.printf("t13.txt=\\"Gateway: %s\\"\\xFF\\xFF\\xFF", Gateway);

const char* dnsIP = WiFi.dnsIP().toString().c_str();
if (strlen(dnsIP) == 0){
    String strDNS = IPtoString(WiFi.dnsIP());
    LCD.printf("t14.txt=\\"DNS IP: %s\\"\\xFF\\xFF\\xFF", strDNS);
}
else
    LCD.printf("t14.txt=\\"DNS IP: %s\\"\\xFF\\xFF\\xFF", dnsIP);
}
else {
    LCD.print("t9.txt=\\"No Network Connection\\"\\xFF\\xFF\\xFF");
}
}

else if(ItemID == id_brightness){
    get_flag = get_BRIGHTNESSNOW;
    LCD.print("get h0.val\\xFF\\xFF\\xFF");
    extendPageTimeout();
}

else if(ItemID == id_voicevol){
    get_flag = get_VOICEVOLNOW;
    LCD.print("get h1.val\\xFF\\xFF\\xFF");
    extendPageTimeout();
}

else if(ItemID == id_dimtime || ItemID == id_alarmperiod || ItemID ==
id_setsnooze)
    extendPageTimeout();

```

```

else
    Console.printf(" Unknown Settings Page ID:%d",ItemID);
}
}

void handleNetworkPage(){

int ItemID;

if (LCD.available()) {
    ItemID = LCD.read();
    Console.printf(" Item ID:%d", ItemID);
    flushLCD();
    beep(1200,50);
    if(ItemID == id_networkback)
        showMainPage(); //back to main page
    else if (ItemID == id_networksave){
        Console.printf(" Network SAVE button Pressed - ");
        get_flag = get_SSID;           // Get values to save
        LCD.print("get t3.txt\xFF\xFF\xFF");
    }
    else if (ItemID == id_connect) {
        Console.printf(" Network CONNNECT button Pressed - ");
        showMainPage();
        connectWIFI();
        //get_flag = get_SSID;           // Get values to save
        //LCD.print("get t3.txt\xFF\xFF\xFF");
    }
    else if (ItemID == id_ssid || ItemID == id_pass || ItemID == id_server ||
ItemID == id_ntp )
        extendPageTimeout();
    else
        Console.printf(" Unknown Network Page ID:%d",ItemID);
}
}

```



```

}
}

void handleDateTimePage(){

int ItemID;

if (LCD.available()) {
    ItemID = LCD.read();
    Console.printf(" Item ID:%d", ItemID);
    flushLCD();
    beep(1200,50);
    if(ItemID == id_datetimeback)
        showMainPage(); //back to main page
    else if(ItemID == id_changedatetime){
        get_flag = get_SETDAY;
        LCD.print("get n2.val\xFF\xFF\xFF");
        //showMainPage(); //back to main page
    }
    else if(ItemID = id_syncntp){
        showMainPage();
        if(wifiStatus == wifi_connected){
            syncNTPtime();
            beep(700,100);
        }
        else
            LCD.print("t4.txt=\\"No Wifi for NTP Sync\\" \xFF\xFF\xFF");
    }
    else
        Console.printf(" Unknown Date Time Page ID:%d",ItemID);
}
}

void handleIncomingNum(){

```

```

int byte1 = 0, byte2 = 0, byte3 = 0, byte4 = 0;
int Num;

delayMicroseconds(100);
if (LCD.available())
  byte1 = LCD.read();
if (LCD.available())
  byte2 = LCD.read();
if (LCD.available())
  byte3 = LCD.read();
if (LCD.available()){
  byte4 = LCD.read();
  Num = byte1 + byte2*256 + byte3*65536 + byte4*16777216;
  if(get_flag == get_RTCYEAR){
    if(Num > 2000 && Num < 2500){
      curr_YEAR = Num;
      get_flag = get_RTCMONTH;
      LCD.printf("get rtc1\xFF\xFF\xFF");
    }
  }
  else if (get_flag == get_RTCMONTH){
    if(Num > 0 && Num < 13){
      curr_MONTH = Num;
      get_flag = get_RTCDAY;
      LCD.printf("get rtc2\xFF\xFF\xFF");
    }
  }
  else if (get_flag == get_RTCDAY){
    if(Num > 0 && Num < 32){
      curr_DAY = Num;
      if(!page_timeout)

```

```

        LCD.printf("t2.txt=\">%02d/%02d/%d\"\\x\\x\\x",curr_DAY,
curr_MONTH, curr_YEAR);
        get_flag = get_RTCHOUR;
        LCD.printf("get rtc3\\x\\x\\x");
    }
}
else if (get_flag == get_RTCHOUR){
    if(Num >= 0 && Num < 25){
        curr_HOUR = Num;
        get_flag = get_RTCMINUTE;
        LCD.printf("get rtc4\\x\\x\\x");
    }
}
else if (get_flag == get_RTCMINUTE){
    if(Num >= 0 && Num < 61){
        curr_MINUTE = Num;
        get_flag = get_RTCSECOND;
        LCD.printf("get rtc5\\x\\x\\x");
    }
}
else if (get_flag == get_RTCSECOND){
    if(Num >= 0 && Num < 61){
        curr_SECOND = Num;
        get_flag = 0; // Clear get_flag
        if(!page_timeout)

LCD.printf("t3.txt=\">%02d:%02d      %02d\"\\x\\x\\x",curr_HOUR,
curr_MINUTE, curr_SECOND);
    }
}
else if (get_flag == get_SETDAY){
    page_timeout = 2;          //Give 2 second for all set date and
time to come in

```

```

    setDay = Num;
    Console.printf("\nSet Day: %d", setDay);
    get_flag = get_SETMONTH;
    LCD.print("get n3.val\xFF\xFF\xFF");
}
else if (get_flag == get_SETMONTH){
    setMonth = Num;
    Console.printf("\nSet Month: %d", setMonth);
    get_flag = get_SETYEAR;
    LCD.print("get n0.val\xFF\xFF\xFF");
}
else if (get_flag == get_SETYEAR){
    setYear = Num;
    Console.printf("\nSet Year: %d", setYear);
    get_flag = get_SETHOUR;
    LCD.print("get n4.val\xFF\xFF\xFF");
}
else if (get_flag == get_SETHOUR){
    setHour = Num;
    Console.printf("\nSet Hour: %d", setHour);
    get_flag = get_SETMINUTE;
    LCD.print("get n5.val\xFF\xFF\xFF");
}
else if (get_flag == get_SETMINUTE){
    setMinute = Num;
    Console.printf("\nSet Minute: %d", setMinute);
    get_flag = get_SETSECOND;
    LCD.print("get n1.val\xFF\xFF\xFF");
}
else if (get_flag == get_SETSECOND){
    setSecond = Num;
    Console.printf("\nSet Second: %d", setSecond);
    LCD.printf("rtc0=%d\xFF\xFF\xFF", setYear);
}

```

```

LCD.printf("rtc1=%d\xFF\xFF\xFF", setMonth);
LCD.printf("rtc2=%d\xFF\xFF\xFF", setDay);
LCD.printf("rtc3=%d\xFF\xFF\xFF", setHour);
LCD.printf("rtc4=%d\xFF\xFF\xFF", setMinute);
LCD.printf("rtc5=%d\xFF\xFF\xFF", setSecond);
showMainPage();
get_flag = 0; // Clear get_flag
}
else if (get_flag == get_BRIGHTNESSNOW){
    get_flag = 0;
    LCD.printf("dim=%d\xFF\xFF\xFF",Num);
}
else if (get_flag == get_VOICEVOLNOW){
    get_flag = 0;
    LCD.printf("volume=%d\xFF\xFF\xFF",Num);
    LCD.printf("play 0,%d,0\xFF\xFF\xFF",audio_testSpeech); // testing
speech
}
else if (get_flag == get_DIMTIME){
    curr_DIMMINS = Num;
    Console.printf(" \nDim Duration in Minutes:%d",curr_DIMMINS);
    get_flag = get_BRIGHTNESS;
    EEPROM.write(EEaddr_dimtime,curr_DIMMINS);
    LCD.print("get h0.val\xFF\xFF\xFF");
}
else if (get_flag == get_BRIGHTNESS){
    curr_BRIGHTNESS = Num;
    Console.printf(" \nScreen Brightness:%d",curr_BRIGHTNESS);
    get_flag = get_VOICEVOL;
    EEPROM.write(EEaddr_bright,curr_BRIGHTNESS);
    LCD.print("get h1.val\xFF\xFF\xFF");
}
else if (get_flag == get_VOICEVOL){

```

```

curr_VOLUME = Num;
Console.printf(" \nVoice Volume:%d",curr_VOLUME);
get_flag = get_ALARM;
EEPROM.write(EEaddr_vol,curr_VOLUME);
LCD.print("get n1.val\xFF\xFF\xFF");
}
else if (get_flag == get_ALARM){
    if (Num >= 0 && Num <= 180) {
        curr_ALARMMINS = Num;
        if (seconds_left > (curr_ALARMMINS*60))
            seconds_left = curr_ALARMMINS*60;        // to make sure the old
alarm is tuned into the new alarm if overtime
        Console.printf(" \nAlarm Duration:%d",curr_ALARMMINS);
    }
    get_flag = get_SNOOZE;
    EEPROM.write(EEaddr_alarm,curr_ALARMMINS);
    LCD.print("get n2.val\xFF\xFF\xFF");
}
else if (get_flag == get_SNOOZE){ // All settings element values
collected
    if (Num >= 0 && Num < 10) {
        curr_SNOOZE = Num;
        Console.printf(" \nSnooze Duration:%d",curr_SNOOZE);
    }
    EEPROM.write(EEaddr_snooze,curr_SNOOZE);
    LCD.printf("dims=%d\xFF\xFF\xFF", curr_BRIGHTNESS);
    EEPROM.commit();
    showMainPage();
}
else {
    Console.printf(" Unknown Incoming Number: %d", Num);
    flushLCD();
}
}

```

```
        flushLCD();
    }
    else {
        Console.printf(" No Incoming Number Received!");
    }
}

void handleIncomingString() {

    delayMicroseconds(100);
    //Console.printf(" get_flag:%d",get_flag);
    if (get_flag == get_SSID){
        receiveSSID();
        get_flag = get_PASS;
        LCD.print("get t4.txt\xFF\xFF\xFF");
    }
    else if (get_flag == get_PASS){
        receivePASS();
        get_flag = get_ServerURL;
        LCD.print("get t6.txt\xFF\xFF\xFF");
    }
    else if (get_flag == get_ServerURL){
        receiveServerURL();
        get_flag = get_NTPURL;
        LCD.print("get t8.txt\xFF\xFF\xFF");
    }
    else if (get_flag == get_NTPURL){
        receiveNTPURL();
        get_flag = 0;
        showMainPage();
    }
    else{
        Console.printf(" Unknown Incoming String !");
    }
}
```

```
        flushLCD();
    }

}

void receiveSSID() {

    char byteIn;
    int n = 0;

    while (LCD.available()) {
        byteIn = LCD.read();
        if(byteIn == 0xFF){
            Wifi_SSID[n] = 0; // Set null for end of string
            flushLCD();
            break;
        }
        Wifi_SSID[n] = byteIn;
        n++;
        if(n > 31){
            Wifi_SSID[32] = 0; // Set null for end of string
            break;
        }
        delay(1);
    }
    Console.printf(" \nSSID: %s",Wifi_SSID);
    writeSSIDToEEPROM();
}

void receivePASS() {

    char byteIn;
    int n = 0;
```



```
while (LCD.available()) {
  byteIn = LCD.read();
  if(byteIn == 0xFF){
    Wifi_PASS[n] = 0; // Set null for end of string
    flushLCD();
    break;
  }
  Wifi_PASS[n] = byteIn;
  n++;
  if(n > 31){
    Wifi_PASS[32] = 0; // Set null for end of string
    break;
  }
  delay(1);
}
Console.printf(" \nPassword: %s",Wifi_PASS);
writePASSToEEPROM();
// EEPROM.commit();
}

void receiveServerURL() {

  char byteIn;
  int n = 0;

  while (LCD.available()) {
    byteIn = LCD.read();
    if(byteIn == 0xFF){
      Wifi_ServerURL[n] = 0; // Set null for end of string
      flushLCD();
      break;
    }
  }
```

```
Wifi_ServerURL[n] = byteIn;
n++;
if(n > 31){
    Wifi_ServerURL[32] = 0; // Set null for end of string
    break;
}
delay(1);
}
Console.printf("\nServer URL : %s",Wifi_ServerURL);
}
```

```
void receiveNTPURL() {

char byteIn;
int n = 0;

while (LCD.available()) {
    byteIn = LCD.read();
    if(byteIn == 0xFF){
        Wifi_NTPURL[n] = 0; // Set null for end of string
        flushLCD();
        break;
    }
    Wifi_NTPURL[n] = byteIn;
    n++;
    if(n > 31){
        Wifi_NTPURL[32] = 0; // Set null for end of string
        break;
    }
    delay(1);
}
Console.printf("\nNTP URL : %s",Wifi_NTPURL);
writeNTPURLToEEPROM();
```

```

EEPROM.commit();
Console.print("\nEEPROM saved");

if(wifiStatus == wifi_connected)
    initNTP();                // Use new NTP URL
}

void showAlarmCountdown(){

    int alarm_progress;

    if (curr_ALARMMINS == 0) // prevent divide by 0 error
        return;

    alarm_progress = (seconds_left*100)/(curr_ALARMMINS*60);

    timerHour = seconds_left/3600;
    timerMins = seconds_left/60 - (timerHour*60);
    timerSecs = seconds_left - (timerMins*60) - (timerHour*60);

    LCD.printf("t6.txt=\">%dh  %02dm  %02ds\%03s", timerHour,
timerMins, timerSecs);
    LCD.printf("j0.val=%02d\%03s", alarm_progress);

}

void flushLCD() {
    delay(1);
    while (LCD.available()) {
        LCD.read();
        delayMicroseconds(100);
    }
}

```

```

    }
}

void beep(int freq, int dura) {
    tone(BUZZPIN, freq, dura);
}

void flashLED(int dura) {

    digitalWrite(LED_BUILTIN, HIGH);
    delay(dura);
    digitalWrite(LED_BUILTIN, LOW);
}

void showMainPage() {
    LCD.printf("page %d\\xFF\\xFF\\xFF", page_main);
    if(occupiedFlag){
        LCD.print("t0.txt=\\\"OCCUPIED\\\"\\xFF\\xFF\\xFF");
        LCD.print("t0.bco=65504\\xFF\\xFF\\xFF");
    }
    else {
        LCD.print("t0.txt=\\\"VACANT\\\"\\xFF\\xFF\\xFF");
        LCD.print("t0.bco=46522\\xFF\\xFF\\xFF");
    }
    if(wifiStatus == wifi_connected)
        LCD.print("p4.pic=9\\xFF\\xFF\\xFF");
    getRTCdatetime();
    page_timeout = 0;
}

void getRTCdatetime(){
    get_flag = get_RTCYEAR;
    LCD.printf("get rtc0\\xFF\\xFF\\xFF");
}

```

```
}

```

```
void extendPageTimeout(){

```

```
    page_timeout = page_timeout + timeout_extension; // add extension period

```

```
    if(page_timeout > timeout_max)

```

```
        page_timeout = timeout_max;

```

```
}

```

```
void check_alarmTimeout(){

```

```
    if (seconds_left > 0)

```

```
        seconds_left--;

```

```
    else { // seconds left has reached 0

```

```
        if (secs%5 == 0){ // Every 5 seconds

```

```
            if (curr_ALARMMINS > 0){

```

```
                LCD.print("t0.txt=\"TURN PATIENT !\"\\xFF\\xFF\\xFF");

```

```
                LCD.print("t0.bco=62286\\xFF\\xFF\\xFF");

```

```
                if(!mute_flag){

```

```
                    beep(1600,200);

```

```
                    LCD.printf("play 0,%d,0\\xFF\\xFF\\xFF",audio_turnPatient); // voice

```

```
alarm to turn patient

```

```
        }

```

```
    }

```

```


```

```


```

```
}

```

```
void storeStatusChg_time(int s){

```

```
    for(int i = 0; i < 7; i++){

```

```
        statusTime_5[i] = statusTime_4[i];

```

```
        statusTime_4[i] = statusTime_3[i];

```

```
        statusTime_3[i] = statusTime_2[i];

```

```

    statusTime_2[i] = statusTime_1[i];
}

statusTime_1[0] = 1;
if(s == 1){
    statusTime_1[1] = 1; // OCCUPIED
}
else if (s == 0){
    statusTime_1[1] = 0; // VACANT
}
statusTime_1[2] = curr_DAY;
statusTime_1[3] = curr_MONTH;
statusTime_1[4] = curr_YEAR;
statusTime_1[5] = curr_HOUR;
statusTime_1[6] = curr_MINUTE;
}

// SDA goes HIGH to LOW with SCL HIGH
void i2c_start() {

    digitalWrite(SCL_PIN, HIGH); // Initially drive SCL HIGH
    pinMode(SCL_PIN, OUTPUT);
    delayMicroseconds(2);
    digitalWrite(SDA_PIN, HIGH); // Initially drive SDA HIGH
    pinMode(SDA_PIN, OUTPUT);
    delayMicroseconds(2);
    digitalWrite(SDA_PIN, LOW); // Now drive SDA LOW
    digitalWrite(SCL_PIN, LOW); // Leave SCL LOW
}

// SDA goes LOW to HIGH with SCL High
void i2c_stop() {
    digitalWrite(SCL_PIN, HIGH); // Drive SCL HIGH
    delayMicroseconds(2);

```

```

digitalWrite(SDA_PIN, HIGH); // then SDA HIGH
delayMicroseconds(2);
pinMode(SCL_PIN, INPUT); // Now let them float
pinMode(SDA_PIN, INPUT); // If pullups present, they'll stay HIGH
}

bool i2c_Write(uint8_t data) {

    bool ackbit = 0;
    digitalWrite(SCL_PIN, LOW); // start SCL pin low
    //shiftOut(SDA_PIN,SCL_PIN,MSBFIRST,data);

    for (int n = 0; n < 8; n++){ // shift out 8 bits MSB first
        digitalWrite(SDA_PIN, !(data & (1 << (7 - n))));
        delayMicroseconds(10);
        digitalWrite(SCL_PIN, HIGH); // Toggle SCL from LOW to HIGH to
LOW
        delayMicroseconds(10);
        digitalWrite(SCL_PIN, LOW);
        delayMicroseconds(10);
    }
    pinMode(SDA_PIN, INPUT); // Set SDA to input for ACK/NAK
    digitalWrite(SCL_PIN, HIGH);
    ackbit = digitalRead(SDA_PIN); // Sample SDA when SCL is HIGH
    digitalWrite(SCL_PIN, LOW);
    digitalWrite(SDA_PIN, LOW); // Leave SDA driven LOW
    pinMode(SDA_PIN, OUTPUT);

    return ackbit;
}

void i2c_WriteTwinByte(uint8_t addrDev, uint8_t addrReg, uint8_t byte1,
uint8_t byte2) {

```

```

i2c_start();
digitalWrite(SCL_PIN, LOW);      // start SCL pin low
i2c_Write(addrDev & 0b11111110); // send in 40h with zero as LSB for
write
i2c_Write(addrReg);
i2c_Write(byte1);
i2c_Write(byte2);
i2c_stop();
}

```

```

void i2c_WriteByte(uint8_t addrDev, uint8_t addrReg, uint8_t byte1) {
i2c_start();
digitalWrite(SCL_PIN, LOW);      // start SCL pin low
i2c_Write(addrDev & 0b11111110); // send in 40h with zero as LSB for
write
i2c_Write(addrReg);
i2c_Write(byte1);
i2c_stop();
}

```

//Read in i2c data, Data byte is output MSB first, SDA data line is  
//valid only while the SCL line is HIGH. SCL and SDA left in LOW state.

```

uint8_t i2c_Read() {

int s = 0;
uint8_t data = 0;
pinMode(SDA_PIN, INPUT);      //Make SDA an input

for (int n = 0; n < 8; n++){    // shift in 8 bits MSB first
digitalWrite(SCL_PIN, HIGH);   // Toggle SCL from LOW to HIGH to
LOW
delayMicroseconds(20);
data = (data << 1) | digitalRead(SDA_PIN);
}
}

```



```

    digitalWrite(SCL_PIN, LOW);
    delayMicroseconds(10);
}

//uint8_t data = shiftIn(SDA_PIN,SCL_PIN,MSBFIRST);
digitalWrite(SDA_PIN, HIGH);      // Output NACK to SDA by pulling
high
pinMode(SDA_PIN, OUTPUT);
digitalWrite(SCL_PIN, HIGH);      // Toggle SCL from LOW to HIGH to
LOW
delayMicroseconds(10);
digitalWrite(SCL_PIN, LOW);
digitalWrite(SDA_PIN, LOW);      // Leave SDA driven LOW
return data;
}

uint8_t i2c_ReadByte(uint8_t addrDev, uint8_t addrReg) {

    i2c_start();
    digitalWrite(SCL_PIN, LOW);    // start SCL pin low
    i2c_Write(addrDev & 0b11111110); // send in 40h with zero as LSB for
write
    i2c_Write(addrReg); // send in addrReg 0x09h
    i2c_start();
    i2c_Write(addrDev | 0b00000001); // send in 40h again with 1 as LSB for
read
    uint8_t data = i2c_Read();
    i2c_stop();

    return data;
}

```

```

bool IOboards_Check() {

    bool OKFlag = true;
    //Check VT8
    //Console.printf("\nChecking VT8 on bus :");
    i2c_start();
    int ackbit1 = i2c_Write(VT8Addr);
    i2c_stop();
    if (ackbit1 == LOW){
        //Console.print(" VT8 board detected");
        //LCD.print("t0.txt=\"VT8 detected\"\xFF\xFF\xFF");
        VT8_existFlag = true;
        //delay(500);
    }
    else {
        Console.print(" VT8 board not found!");
        //LCD.print("t0.pco=63488\xFF\xFF\xFF"); // RED TEXT colour
        LCD.print("t0.bco=63488\xFF\xFF\xFF");
        LCD.print("t0.txt=\"VT8 board not found!\xFF\xFF\xFF");
        OKFlag = false;
        delay(1000);
    }

    //Check HZ16
    //Console.printf("\nChecking HZ16 on bus :");
    i2c_start();
    int ackbit2 = i2c_Write(HZ16Addr);
    i2c_stop();
    if (ackbit2 == LOW) {
        //Console.print(" HZ16 board detected");
        //LCD.print("t0.txt=\"HZ16 detected\"\xFF\xFF\xFF");
        HZ16_existFlag = true;
        //delay(500);
    }
}

```

```

    }
    else {
        Console.print(" HZ16 board not found!");
        //LCD.print("t0.pco=63488\xFF\xFF\xFF"); // RED TEXT colour
        LCD.print("t0.bco=63488\xFF\xFF\xFF");
        LCD.print("t0.txt=\\"HZ16 board not found!\\" \xFF\xFF\xFF");
        OKFlag = false;
    }
    return OKFlag;
}

void init_HZ16() {

    i2c_WriteTwinByte(HZ16Addr, HZ16_DIRA, 0b11111111, 0b11111111);
    // Set all pins to INPUT to not interfere with sensing
    i2c_WriteTwinByte(HZ16Addr, HZ16_GPIOA, 0b11111111, 0b11111111);
    Console.print("\nHZ16 board initialized");
}

int scan_bedArray() { //returns number of points under pressure

    uint8_t patternA = 0b11111110;
    uint8_t patternB = 0b11111110;
    uint8_t row;
    int yloc = 400; // Placement on LCD screen
    int totalNum = 0;
    int n, m;

    //Console.print("\n\n==== SCANNING BED =====");
    digitalWrite(LED_BUILTIN, HIGH);
    i2c_WriteTwinByte(HZ16Addr, HZ16_GPIOA, 0b00000000, 0b00000000);
    // All pins preset to low but unconnected (INPUT)

```

```

//Scan patient from feet to head
for (int i = 0; i < 8; i++){          // Bottom half (8) of bed
    i2c_WriteTwinByte(HZ16Addr, HZ16_DIRA, patternA, 0b11111111);
    delayMicroseconds(10);          // delay a bit for line capacitance
    patternA = (patternA << 1) | 0b00000001;
    row = i2c_ReadByte(VT8Addr,VT8_GPIO);
    if (row == 0) {                  // if entire line zero suspect ghost, read
again
        //beep(850,100);
        //Console.print("PortA Error");
        row = i2c_ReadByte(VT8Addr,VT8_GPIO);
    }
    n = draw_bedRow(row,yloc);
    bedArray[i] = row;              // Store row
    totalNum = totalNum + n;        // Sum no. of lit points
    yloc = yloc - 17; // 17 pixels vertical separation of dots on LCD
    delayMicroseconds(10); // delay to prevent reading reflections
}
for (int j = 0; j < 8; j++){        // top half (8) of bed
    i2c_WriteTwinByte(HZ16Addr, HZ16_DIRA, 0b11111111, patternB);
    delayMicroseconds(10);          // delay a bit for line capacitance
    patternB = (patternB << 1) | 0b00000001;
    row = i2c_ReadByte(VT8Addr,VT8_GPIO);
    if (row == 0) {                  // if entire line zero suspect ghost, read
again
        //beep(850,100);
        //Console.print("PortB Error");
        row = i2c_ReadByte(VT8Addr,VT8_GPIO);
    }
    if(j > 5){                       // Ignore first two line near headboard for pillow
        draw_nullbedRow(row,yloc);
        m = 0;
    }
}

```

```

else
    m = draw_bedRow(row,yloc);
    bedArray[j + 8] = row;          // Store row
    totalNum = totalNum + m;       // Sum no. of lit points
    yloc = yloc - 17; //17 pixels vertical separation of dots on LCD
    delayMicroseconds(10);        // delay to prevent reading reflections
}
digitalWrite(LED_BUILTIN, LOW);

return totalNum;
}

int draw_bedRow(uint8_t row, int yloc ){

    int xloc = 87;                 // Placement on LCD screen
    int count = 0;

    for (int i = 7; i >= 0; i--) {
        if(bitRead(row,i)==1)
            LCD.printf("fill %d,%d,10,10,2016\xFF\xFF\xFF", xloc, yloc);          //
GREEN
        else{
            LCD.printf("fill %d,%d,10,10,63488\xFF\xFF\xFF", xloc, yloc);
//RED - Point Pressed
            count++;
        }
        xloc = xloc + 17; //17 pixels horizontal separation of dots on LCD
    }

    return count;
}

int draw_nullbedRow(uint8_t row, int yloc ){

```

```

int xloc = 87;           // Placement on LCD screen
int count = 0;

for (int i = 7; i >= 0; i--) {
  if(bitRead(row,i)==1)
    LCD.printf("fill %d,%d,10,10,2016\xFF\xFF\xFF", xloc, yloc);      //
  GREEN
  else{
    LCD.printf("fill %d,%d,10,10,31\xFF\xFF\xFF", xloc, yloc);
  //BLUE - Null Point Pressed
    count++;
  }
  xloc = xloc + 17; //17 pixels horizontal separation of dots on LCD
}

return count;
}

void powerON_bedArray() {
  digitalWrite(IOPW_PIN, HIGH);
}

void powerOFF_bedArray() {
  digitalWrite(IOPW_PIN, LOW);
}

void hardReset_bedArray(){
  powerOFF_bedArray();
  delay(1500);
  powerON_bedArray();
  delay(300);
}

```

```

i2c_WriteByte(VT8Addr, VT8_IPOL, 0b00000000); // Make sure input
polarity register is not inverted
i2c_WriteByte(HZ16Addr, HZ16_IOCON1, 0b00000000); // Make sure
IOCON register is all zeroised
i2c_WriteByte(HZ16Addr, HZ16_IOCON2, 0b00000000); // Make sure
IOCON register is all zeroised
init_HZ16();
}

```

```

void checkBedStatus() {

```

```

    uint8_t val1 = i2c_ReadByte(VT8Addr,VT8_IPOL);
    uint8_t val2 = i2c_ReadByte(HZ16Addr,HZ16_IOCON1);
    uint8_t val3 = i2c_ReadByte(HZ16Addr,HZ16_IOCON2);

```

```

    int totalVal = val1 + val2 + val3;

```

```

    if (totalVal > 0){

```

```

        BedErrFlag = true;

```

```

        LCD.print("t5.pco=63488\xFF\xFF\xFF");

```

```

        LCD.print("t5.txt=\"Resetting Bed Array...\xFF\xFF\xFF");

```

```

        beep(850,300);

```

```

        hardReset_bedArray();

```

```

        LCD.print("t5.txt=\" \xFF\xFF\xFF");

```

```

    }

```

```

}

```

```

void connectWIFI(){

```

```

    char *ssid;

```

```

    char *pass;

```

```

    ssid = &Wifi_SSID[0];

```

```

    pass = &Wifi_PASS[0];

```

```

if (WiFi.status() == WL_CONNECTED) // if connected, disconnect first
  WiFi.disconnect();

wifiStatus = wifi_connecting;
WiFi.setHostname(HostName);
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, pass);
Console.printf("\nConnecting to WiFi %s ...", Wifi_SSID);
LCD.printf("t4.txt=\\"Connecting to WiFi %s ...\\"\\xFF\\xFF\\xFF", Wifi_SSID);

}

void checkwifiConnected() {

if (WiFi.status() == WL_CONNECTED) {
  wifiStatus = wifi_connected;
  LCD.print("p4.pic=9\\xFF\\xFF\\xFF");
  LCD.printf("t4.txt=\\"Connected to %s\\"\\xFF\\xFF\\xFF", Wifi_SSID);
  beep(900,100);
  Console.print("\n\nLocal IP: ");
  Console.println(WiFi.localIP());
  Console.print("NetMask: ");
  Console.println(WiFi.subnetMask());
  Console.print("Gateway IP: ");
  Console.println(WiFi.gatewayIP());
  Console.print("DNS: ");
  Console.println(WiFi.dnsIP());
  initNTP();
  HTTPserver.begin();
  Console.print("\nHTTP Server Started");
}

```



```

}

void checkwifiDisconnected() {

    if (WiFi.status() != WL_CONNECTED) {
        wifiStatus = wifi_down;
        LCD.print("p4.pic=8\xFF\xFF\xFF");
        LCD.print("t4.txt=\\"Wifi Network Down\\" \xFF\xFF\xFF");
        beep(800,200);
    }
}

void writeSSIDtoEEPROM()
{
    byte len = strlen(Wifi_SSID);
    // Console.printf("\nSSID: %d bytes", len);
    EEPROM.write(EEaddr_ssid, len);
    for (int i = 0; i < len; i++)
        EEPROM.write(EEaddr_ssid + 1 + i, Wifi_SSID[i]);
}

void writePASSToEEPROM()
{
    byte len = strlen(Wifi_PASS);
    // Console.printf("\nPassword: %d bytes", len);
    EEPROM.write(EEaddr_pass, len);
    for (int i = 0; i < len; i++)
        EEPROM.write(EEaddr_pass + 1 + i, Wifi_PASS[i]);
}

void writeNTPURLtoEEPROM()

```

```

{
    byte len = strlen(Wifi_NTPURL);
    //Console.printf("\nNTP URL: %d bytes", len);
    EEPROM.write(EEaddr_ntp, len);
    for (int i = 0; i < len; i++)
        EEPROM.write(EEaddr_ntp + 1 + i, Wifi_NTPURL[i]);
}

void readSSIDFromEEPROM()
{
    int newStrLen = EEPROM.read(EEaddr_ssid);
    //Console.printf("\nEESSID: %d bytes", newStrLen);
    for (int i = 0; i < newStrLen; i++)
        Wifi_SSID[i] = EEPROM.read(EEaddr_ssid + 1 + i);
    Wifi_SSID[newStrLen] = 0;           // add back null at the end
}

void readPASSFromEEPROM()
{
    int newStrLen = EEPROM.read(EEaddr_pass);
    //Console.printf("\nEEPASSWORD: %d bytes", newStrLen);
    for (int i = 0; i < newStrLen; i++)
        Wifi_PASS[i] = EEPROM.read(EEaddr_pass + 1 + i);
    Wifi_PASS[newStrLen] = 0;         // add back null at the end
}

void readNTPURLFromEEPROM()
{
    int newStrLen = EEPROM.read(EEaddr_ntp);
    //Console.printf("\nEE-NTP: %d bytes", newStrLen);
    for (int i = 0; i < newStrLen; i++)
        Wifi_NTPURL[i] = EEPROM.read(EEaddr_ntp + 1 + i);
    Wifi_NTPURL[newStrLen] = 0;       // add back null at the end
}

```

```

}

String IPtoString(const IPAddress& address){
    return String() + address[0] + "." + address[1] + "." + address[2] + "." +
address[3];
}

void syncNTPtime() {

    struct tm timeinfo;
    if (!getLocalTime(&timeinfo)) {
        Console.println("\nNTP Sync failed");
        if(!page_timeout){
            LCD.print("t4.txt=\nNTP Sync failed\\"\xFF\xFF\xFF");
            return;
        }
    }
}

//Console.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");

int timeYear = timeinfo.tm_year + 1900;
int timeMonth = timeinfo.tm_mon + 1;
int timeDay = timeinfo.tm_mday;
int timeHour = timeinfo.tm_hour;
int timeMins = timeinfo.tm_min;
int timeSecs = timeinfo.tm_sec;

Console.printf("\nNTP Year : %d ",timeYear);
Console.printf(" Month : %d ",timeMonth);
Console.printf(" Day : %d ",timeDay);
Console.printf(" Hour : %d ",timeHour);
Console.printf(" Mins : %d ",timeMins);
Console.printf(" Secs : %d ",timeSecs);

```

```

LCD.printf("rtc0=%d\\xFF\\xFF\\xFF", timeYear);
LCD.printf("rtc1=%d\\xFF\\xFF\\xFF", timeMonth);
LCD.printf("rtc2=%d\\xFF\\xFF\\xFF", timeDay);
LCD.printf("rtc3=%d\\xFF\\xFF\\xFF", timeHour);
LCD.printf("rtc4=%d\\xFF\\xFF\\xFF", timeMins);
LCD.printf("rtc5=%d\\xFF\\xFF\\xFF", timeSecs);

if(!page_timeout){
  LCD.print("t4.txt=\\nNTP Sync Sucess\\n\\xFF\\xFF\\xFF");
}
}

void initNTP() {

if (strlen(Wifi_NTPURL) == 0){
  configTime(gmtOffset_sec, daylightOffset_sec, defaultNTPserver);
  Console.printf("\\nNTP %s initialised", defaultNTPserver);
}
else {
  configTime(gmtOffset_sec, daylightOffset_sec, Wifi_NTPURL);
  Console.printf("\\nNTP %s initialised", Wifi_NTPURL);
}
}

void CheckHTTPrequests() {

WiFiClient client = HTTPserver.available(); //Checking for incoming clients
int n = 0;
//String header; // for future buttons (header.indexOf)

if(client) {

```

```

Console.println("\nNew HTTP client:");
String currentLine = "";           //Storing the incoming data in the string
while (client.connected()) {
    n++;
    if(n > 100)
        break;
    if (client.available()) {      //if there is some client data available
        n = 0; // Reset counter
        char c = client.read();    //read a byte
        Console.print(c);
        if (c == 'G'){
            char d = client.read();
            Console.print(d);
            if (d == 'E'){
                char e = client.read();
                Console.print(e);
                if (e == 'T'){
                    char f = client.read();
                    Console.print(f);
                    if (f == ' '){
                        char g = client.read();
                        Console.print(g);
                        if(detectPageRequest(client,g)){
                            break;
                        }
                    }
                }
            }
        }
    }
    if (c == '\n') {              //check for newline character,
        if (currentLine.length() == 0) { //if line is blank it means its the end of
the client HTTP request
            displayMainPage(client);

```

```

        Console.print("Main Webpage Sent!");
        break;                //Going out of the while loop
    }
    else {
        currentLine = "";    //if got a newline, then clear currentLine
    }
}
else if (c != '\r') {
    currentLine += c;        //if got anything else but a carriage return
character
}
}
} // End of While Loop
//client.stop();
Console.printf("\nClient disconnected. n = %d \n", n);
}
}

```

```

bool detectPageRequest(WiFiClient client, char c){

```

```

    if (c == '/'){
        char e = client.read();
        Console.print(e);
        if (e == 'C'){
            char f = client.read();
            Console.print(f);
            if (f == 'L'){
                char g = client.read();
                Console.print(g);
                if (g == 'R'){ // Clear Nurse Flag detected
                    callNurseFlag = false;
                    Console.print("\nNurse Alert Cleared\n");
                }
            }
        }
    }
}

```

```

        client.println("<html><head><meta    http-equiv=\"Refresh\"content='0;
url=\\\"\\\">");
        client.println("<link rel=\"icon\" href=\"data:,\"></head></html>");
        client.println();
        Console.print("Redirect Webpage Sent!");
        return true;
    }
}
}
}
return false;
}

```

```
void displayMainPage(WiFiClient client){
```

```

    client.println("HTTP/1.1 200 OK");
    client.println("Content-type:text/html");
    client.println("Connection: close");
    client.println();
    client.println("<!DOCTYPE html><html>");
    client.println("<html lang=\"en\">");
    client.println("<head><meta                charset=\"UTF-8\"><meta
name=\"viewport\"content=\"width=device-width, initial-scale=1.0\">"); //
make web page reponsive to different device widths
    client.println("<meta http-equiv=\"Refresh\" content=\"10\">");
    client.println("<link rel=\"icon\" href=\"data:,\">>"); // prevent requests
on the favicon.
    client.println("<style>");
    client.println(".button    {padding:    15px    25px;font-size:18px;text-
align:center;cursor:pointer;outline:none;color:#fff;background-
color:dodgerblue;border:none;border-radius:15px;box-shadow:0 9px #999;}");
    client.println(".button:hover {background-color:lightblue}");

```

```

client.println(".button:active {background-color:lightblue;box-shadow:0 5px
#666;transform:translateY(4px)}");
client.println(".flex-container{display:flex;flex-direction: row;}");
client.println(".flex-container > div {border-radius:5px;width:15px;
height:15px; margin:5px; border:3px solid black;background-
color:lightblue}</style>");
client.println("<title>Instrumented Bed</title>");
client.println("<body style=\"background-color:#edff7;\">");

client.println("<TABLE width=100%>");
client.println("<COL width=10%> <COL width=30%> <COL width=60%>");
client.println("<TR>");
client.println("<TD></TD>"); // R1 C1 Skipped
client.println("<TD></TD> <TD>"); // R1 C2 Skipped
client.printf("<h3 style=\"margin:0px;text-align:right;white-
space:nowrap;padding-
right:5em;\">%02d/%02d/%d %02d:%02d</h3></TD>", curr_DAY,
curr_MONTH, curr_YEAR, curr_HOUR, curr_MINUTE); // R1 C3
client.println("</TR><TR>");
client.println("<TD></TD>"); // R2 C1 Skipped
client.println("<TD>"); // R2 C2
client.printf("<h1>Instrumented Bed System #%d</h1>", bedNo);
client.println("</TD> <TD>"); // R2 C3
if(callNurseFlag) {
client.println("<h1 style=\"margin:0px;color:red\">Nurse button
Pressed</h1>");
client.println("<button
type=\"button\"class=\"button\"onclick=\"window.location.href='CLR';\">CL
EAR</button>");
}
client.println("</TD></TR>");
client.println("<TR>"); // R3
client.println("<TD></TD>"); // R3 C1 Skipped

```



```

client.println("<TD>"); // R3 C2
client.println("<br> <div style=\"border-
radius:10px;width:230px;height:20px;text-
align:center;padding:5px;background-color:Teal;border:3px solid
#000;outline-color:lightblue;outline-style:double\">");
client.print("<h2 style=\"margin:0px;color:white\"> Head Board </h2>
</div>");
for(int i=15; i>-1; i--){ // scrolling horizontal lines
client.println("<div class=flex-container>");
for(int j=7; j>-1; j--){ // scrolling left to right
if(i>13){
if(bitRead(bedArray[i], j) == 0)
client.println("<div style=background-color:blue></div>"); // draw
blue box
else
client.println("<div></div>"); // draw green box
}
else{
if(bitRead(bedArray[i], j) == 0)
client.println("<div style=background-color:red></div>"); // draw red
box
else
client.println("<div></div>"); // draw green box
}
}
client.println("</div>");
}
client.println("</TD>");
client.printf("<TD valign=top>"); // R3 C3
client.printf("<table><tr>");
if(occupiedFlag){

```

```

        client.println("<br><div
                                style=\"border-
radius:10px;width:200px;height:30px;text-
align:center;padding:10px;border:3px
                                solid
                                #000;outline-
color:OliveDrab;outline-style:double\">");
        client.print("<h2 style=\"margin:0px;color:Maroon\"> OCCUPIED </h2>
</div>");

        client.printf("<br><h2>Time Left: %dh %02dm %02ds </h2>", timerHour,
timerMins, timerSecs);

        if(timerHour + timerMins + timerSecs == 0)
            client.print("<br><h2 style=\"margin:0px;color:purple\">Please turn
patient!</h2>");
        }
        else{
            client.println("<br><div
                                style=\"border-
radius:10px;width:200px;height:30px;text-
align:center;padding:10px;border:3px
                                solid
                                #000;outline-
color:OliveDrab;outline-style:double\">");
            client.print("<h2 style=\"margin:0px;color:DarkSlateBlue\"> VACANT
</h2> </div>");
        }
        client.println("</tr><tr>");
        client.println("<br><h3>BED STATUS HISTORY:</h3>");
        displayBedStatusHistory(client);
        client.println("</tr></table></TD></TR>");
        client.println("</TABLE>");
        client.println("</body>");
        client.println();
    }

void displayBedStatusHistory(WiFiClient client){

    if(statusTime_1[0] == 1){

```

```

if(statusTime_1[1] == 0){
    client.printf("%02d/%02d/%d          %02d:%02d          -
VACANT",statusTime_1[2],statusTime_1[3],statusTime_1[4],statusTime_1[5
],statusTime_1[6]);
}
else if(statusTime_1[1] == 1){
    client.printf("%02d/%02d/%d          %02d:%02d          -
OCCUPIED",statusTime_1[2],statusTime_1[3],statusTime_1[4],statusTime_1
[5],statusTime_1[6]);
}
}

```

```

if(statusTime_2[0] == 1){
    if(statusTime_2[1] == 0){
        client.printf("<br>%02d/%02d/%d          %02d:%02d          -
VACANT",statusTime_2[2],statusTime_2[3],statusTime_2[4],statusTime_2[5
],statusTime_2[6]);
    }
    else if(statusTime_2[1] == 1){
        client.printf("<br>%02d/%02d/%d          %02d:%02d          -
OCCUPIED",statusTime_2[2],statusTime_2[3],statusTime_2[4],statusTime_2
[5],statusTime_2[6]);
    }
}
}

```

```

if(statusTime_3[0] == 1){
    if(statusTime_3[1] == 0){
        client.printf("<br>%02d/%02d/%d          %02d:%02d          -
VACANT",statusTime_3[2],statusTime_3[3],statusTime_3[4],statusTime_3[5
],statusTime_3[6]);
    }
    else if(statusTime_3[1] == 1){

```

```

        client.printf("<br>%02d/%02d/%d          %02d:%02d          -
OCCUPIED",statusTime_3[2],statusTime_3[3],statusTime_3[4],statusTime_3
[5],statusTime_3[6]);
    }
}

if(statusTime_4[0] == 1){
    if(statusTime_4[1] == 0){
        client.printf("<br>%02d/%02d/%d          %02d:%02d          -
VACANT",statusTime_4[2],statusTime_4[3],statusTime_4[4],statusTime_4[5
],statusTime_4[6]);
    }
    else if(statusTime_4[1] == 1){
        client.printf("<br>%02d/%02d/%d          %02d:%02d          -
OCCUPIED",statusTime_4[2],statusTime_4[3],statusTime_4[4],statusTime_4
[5],statusTime_4[6]);
    }
}

if(statusTime_5[0] == 1){
    if(statusTime_5[1] == 0){
        client.printf("<br>%02d/%02d/%d          %02d:%02d          -
VACANT",statusTime_5[2],statusTime_5[3],statusTime_5[4],statusTime_5[5
],statusTime_5[6]);
    }
    else if(statusTime_5[1] == 1){
        client.printf("<br>%02d/%02d/%d          %02d:%02d          -
OCCUPIED",statusTime_5[2],statusTime_5[3],statusTime_5[4],statusTime_5
[5],statusTime_5[6]);
    }
}
}
}
}

```