WIRELESS CONTROLLED ROBOT WITH HAND FOR DANGEROUS TASK

BY

KIAT WEI PAU

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNLOGY (HONS)

COMPUTER ENGINEERING

Faculty of Information and Communication Technology

(Perak Campus)

May 2015

DECLARATION OF ORIGINALITY

I declare that this report entitled "**WIRELESS CONTROLLED ROBOT WITH HAND FOR DANGEROUS TASK**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature	:	___________________________
Name	:	KIAT WEI PAU
Date	:	31th AUGUST 2015

ACKNOWLEDGEMENT

First of all, a very deep appreciation to my project supervisor, Mr. Lee Wai Kong for his patient and enthusiasm when guide me for this project and also the opportunity he gives allow me to learn how to build a real robot. In the past time attends the lecture of Mr Lee Wai Kong about the embedded system, I had been exposed with some knowledge about how to build the system controlling hardware using microcontroller. By this knowledge, I had able to write a set of system to control the robot. Also thank to Mr. Leong Chun Farn for his knowledge transfer when attending his lecture about the computer vision and image processing past time. With the knowledge and technique taught by Mr. Leong, I was able to write a simple fire detection application using image processing method for this project.

Last but not least, thanks FICT for offering me this final year project. Throughout this project, I had been trained to handle whole project, solve for tasks, learn new microcontroller platform, how to write a proper technical report, how to present the work and etc.

Abstract

In real life, the robotic is design in such a way that it can helps to assists and entertain in human daily life, for example robotic vacuum is designed for daily household cleaning, robotic pets designed to entertain human within a safety manner, robotic hand or leg for disable designed for helping disable to back to normal life and etc. Robotic also used in medical purpose especially in surgery and physical therapy. For surgery purpose, the robotic is used to assist surgeon to do a very details level, specific and difficult surgery by using remote surgery hand and the wound area will be even smaller compare with using normal hand while doing the surgery. While for physical therapy, patient is assists by robotic to do some rehabilitation exercise. One of the famous used in physical therapy robotic namely interactive motion technologies robotic.

As mention in above task are operating in safe environment and condition rather. However, a dangerous task in unsafe environment may bring harm to a human body or may cause harm to public. In this case, a robot helper is needed to assists human to complete the dangerous task, for example bomb disposal robot, rescue robot, fire detection robot and etc.

Thus, we propose to develop a low cost mobile robot prototype to perform some dangerous task. Throughout this project, a mobile robot is developed and is able to alert user when detecting a fire. The robot consists of a 4 wheels platform with full robotic hand structure attached to it. Video camera is used for video streaming so that can performs fire detection and alert user when detecting a fire. The robotic car movement is controlled by joystick wirelessly and the robotic hand can be controlled using wireless hand glove.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

RF	Radio Frequency
IR	Infrared
DC	Direct Current
Li-Po	Lithium Polymer
USB	Universal Serial Bus
ICSP	In-Circuit Serial Programming
PC	Personal Computer
LCD	Liquid Crystal Display
PWM	Pulse Width Modulation
UART	Universal Asynchronous Receiver-Transmitter
SPI	Serial Peripheral Interface
IDE	Integrated Development Environment
ADC	Analog-to-digital Converter
DOF	Degree of Freedom

Chapter 1 Introduction

1.1 Problem Statement

Currently, commercialize robot are mainly for safe task that is not include in any destruction crisis and may not bring harm to the robot which can be consider as not designed for being destroy. We may need to think of the dangerous task is always bring destruction and the robot may only use for once. However, currently the production cost of a robot that designed purposely for dangerous task is very expensive, thus, it is not appropriate to be considering as using for one time only. Besides that, a robot for dangerous must be designed with time sensitive manner, which means that it should be more timely precise in doing a task and it should be task specific. As a conclusion, there is very high demand in producing a robot that is low production cost and perform a dangerous task in time sensitive manner.

1.2 Background and motivation

The technologies in industry are glowing sharply in this recent two decade. As glowing day by day, more and more electronic products developed are more advanced than the olden day. At the same time, the robotic industry is demand and desire to produce a robot that’s able to catch up with technologies and at the meanwhile produced more and more robots that are useful to the public, especially for the high risk work task. Due to technologies are getting higher, the development fees also increase with it. Thus, the robots that produced that equipped with high technologies are getting higher in price.

In this project, a mobile robot designed to perform the dangerous task will be discussed. Dangerous task refers to any task that may bring harm to human body while human try to solving the task.

However, to produce a mobile robot for dangerous task, it should be low price in development fee and with high technologies integrated. If it is consider as manufactured for one time purpose or one time use, then it should be as low cost in production fee as well. However, currently commercialize robotics are very expensive and consumers are unaffordable to buy a single of it.

1.3 Project Objective

In this project, the objective is to design, implement and build a low cost mobile robot for dangerous task. There are several sub-objectives need to accomplish in order to successfully achieve our target which are as below:

1. Define, design and construct the structure of the robot
2. Define the grasping mechanism of the robot hand
3. Define the moving mechanism of the robotic car
4. Define the control mechanism of the remote controlling for the mobile robot
5. Construct the RF remote controlling for the robot

By the end of this project, a mobile robot with hand is developed and it can be control by joystick and sensor glove wirelessly. The mobile robot is able to capture the environment video using camera mounted on top of the mobile robot and send it back to user's laptop wirelessly. An image processing based fire detection application will be developed using the video received from the camera.

Chapter 2 Literature Review

2-1 Robotic Car

A research on three omni-directional wheels mobile robot [1] has been done. The physical structure consists of 3 wheels and the wheels are coupled with one motor each and are tie to a single center. Since all the wheels coupled with motor are tie to a single center, each wheel will have 120 degree distance with the other 2 wheels, assume that the length of end to end from wheel to motor are same among 3 wheels, it is forming a equilateral triangle as shown in below diagram.

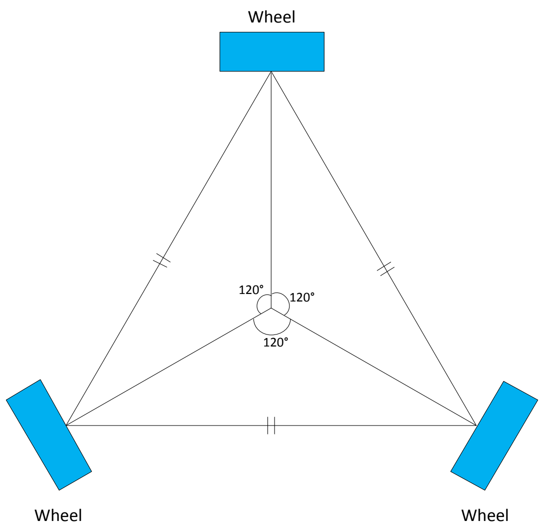


Figure 2-1-1: Omni-directional robotic car conceptual view

To control a movement to a desire direction, three wheels are rotate with specific velocity in order to allow the mobile robot towards the direction wanted. From the research paper, with an example the mobile robot desire to go to α degree direction and the following diagram shows the graphical description of control the velocity of each wheel in order for the mobile robot moving to degree direction.

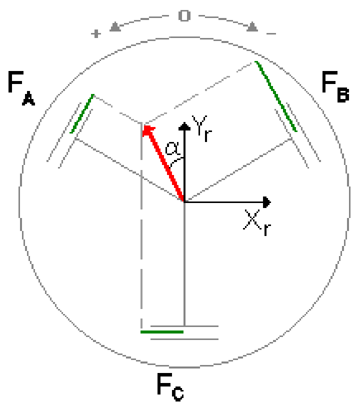


Figure 2-1-2: Graphical view of robotic car motion

The green line shown in the diagram above indicate the length of velocity, the longer line the, higher velocity while the shorter line, the lower velocity. With 3 motor denoted as F_A, F_B and F_C and velocity as linear velocity of the robot, following formula shows how researcher identify the length of velocity of each wheel.

$$F\,A = velocity \cdot \cos(150 - DesiredDirection)$$
$$F\,B = velocity \cdot \cos(30 - DesiredDirection)$$
$$F\,C = velocity \cdot \cos(270 - DesiredDirection)$$

From this research, a series of calculation need to be taken out before any of the development and thus, it is not very suitable for a short period of development time as we plan to do so in this project.

Another type of locomotion based has been come out that is by using 2 wheels [2]. The robot is able to perform self-balancing even with only 2 wheels. Following diagram shows the architecture view of the robot design by the researcher.

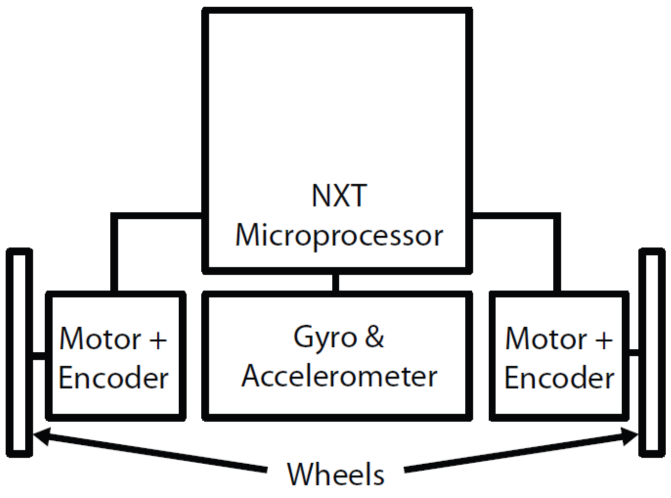


Figure 2-1-3: Architecture view of 2 wheels self-balancing car

Based on the diagram, we can investigate that extra hardware component are needed by researcher to perform balancing for the robot which are gyroscope and accelerometer. Furthermore, this design require some complex mathematic proven before any of the development, including torque applied to the DC-motors, motion of the wheels, the static frictional force produced when the wheels contact with the surface for the robot drives on and etc.

A research on robot with six-wheels has been come out by Viboon Sangveraphunsiri and Mongkol Thianwiboon [3]. This robot consists of six wheels as locomotion based with each side of three wheels. The steering mechanism, which is the way the robot turn directions is by putting 4 steering joint in the rear and front wheels. Following diagram shows the mechanical view of the six-wheel mobile robot developed by the researchers.

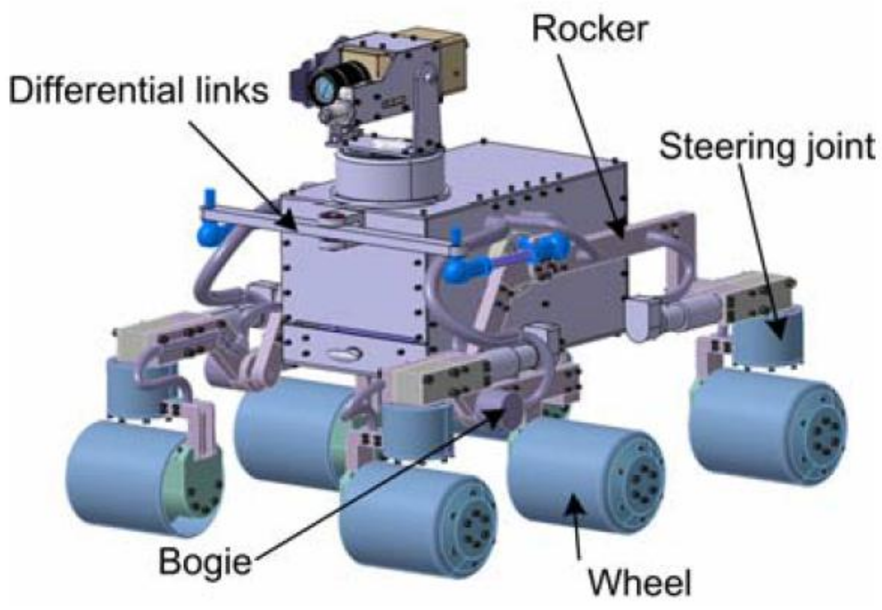


Figure 2-1-4: Mechanical view of the six-wheel mobile robot

An exclusive view of this design is that it is equipped with suspension in which to allow one wheel of the robot can be lifted vertically while other wheels remain in contact with the ground. This is very useful when mobile robot run through an extreme surface area, for example climbing up a slope, traversing over a ditch, traversing over obstacles, climbing upstairs and etc. Following diagrams show the view on how the six-wheel robot performs such action as stated.

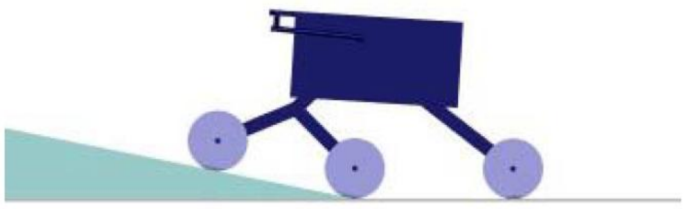


Figure 2-1-5: Six-wheel mobile robot climbs up a slope

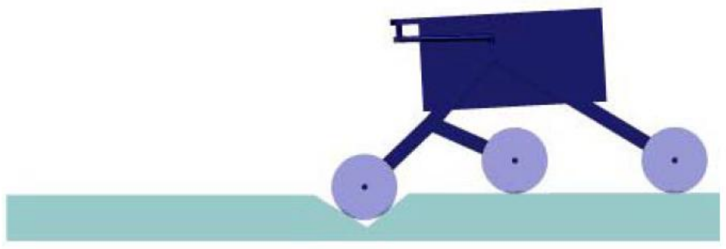


Figure 2-1-6: Six-wheel mobile robot traversing over a ditch

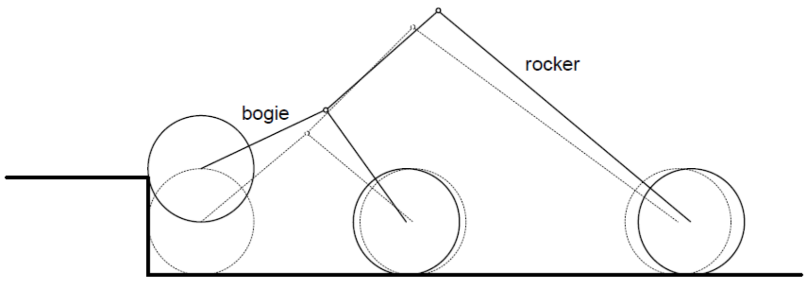


Figure 2-1-7: Six-wheel mobile robot climbs upstairs

However, this design requires extra cost for heavy components use and extra time to do the development since majority of the time wasted in construction of the robot. Furthermore, the wheels in the middle seem wasting resources since there is nothing more than using it for balance the robot when traversing an uneven surface. Therefore, we proposed that using four-wheels as our locomotion based to shorten our development time and more saving cost.

2.2 Robotic Hand

For a robotic hand with multiple fingers, basically the actuation structure of the fingers can be classified into 2 types, the underactuated structure and fully actuated. For underactuated structure, the design used to coupled more than one DOF of the finger associate to a single actuator while fully actuated structure define to associate one degree of freedom to one actuator. There are pros and cons for each structure. The underactuated structure is small in size since mount one actuator to associate several DOFs, lower weight, easy to develop and lower cost of design since using lesser actuator. But in terms of accuracy, since using lesser actuator to control more DOFs, the DOFs of the finger would be limited. For fully actuated structure, it is benefit that it can accurately achieve all the DOFs of the finger since using one actuator to control one DOF. However, in another side of view, increasing in size, weight, complexity of control, development time and cost of development would restrict researchers to do any further development.

For the hand structure, three fingers robotic hand [4] has been developed by researchers that can perform a set of challenges task set by the researchers. Following diagram shows the structure of the robotic hand that been developed.

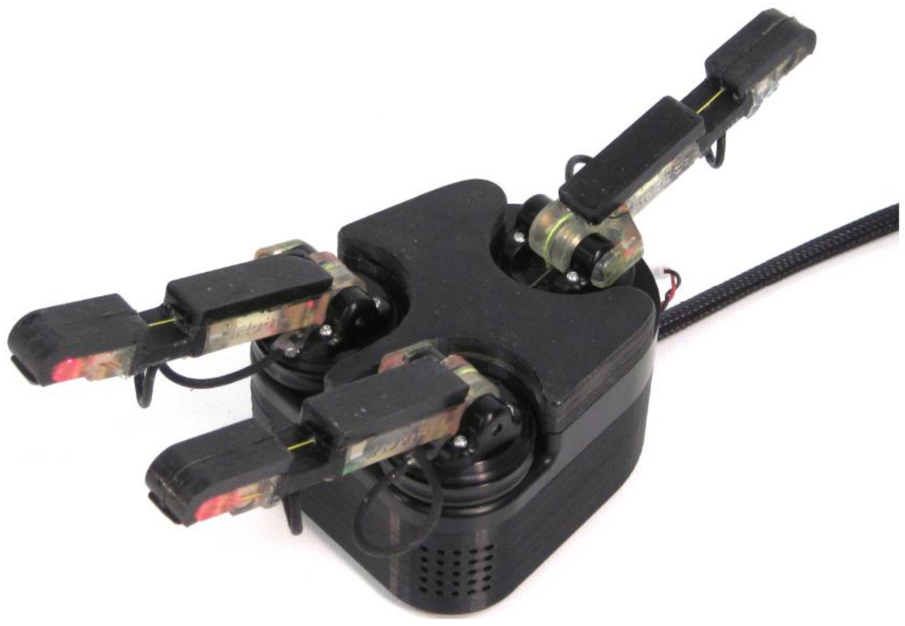


Figure 2-2-1: Structure of three fingers robotic hand

There is 1 finger represent the thumb of the robotic hand. Total of 5 actuators used by this robotic hand with 2 used by thumb since it is a fully actuated structure that consisting of 2 DOFs, 1 actuator each for second finger and third finger to control the underactuated structure of finger that consists of 2 DOFs and the last actuator used to control the horizontal distance of the second finger third finger. The diagram below shows the actuation scheme of the three fingers robotic hand discussed.

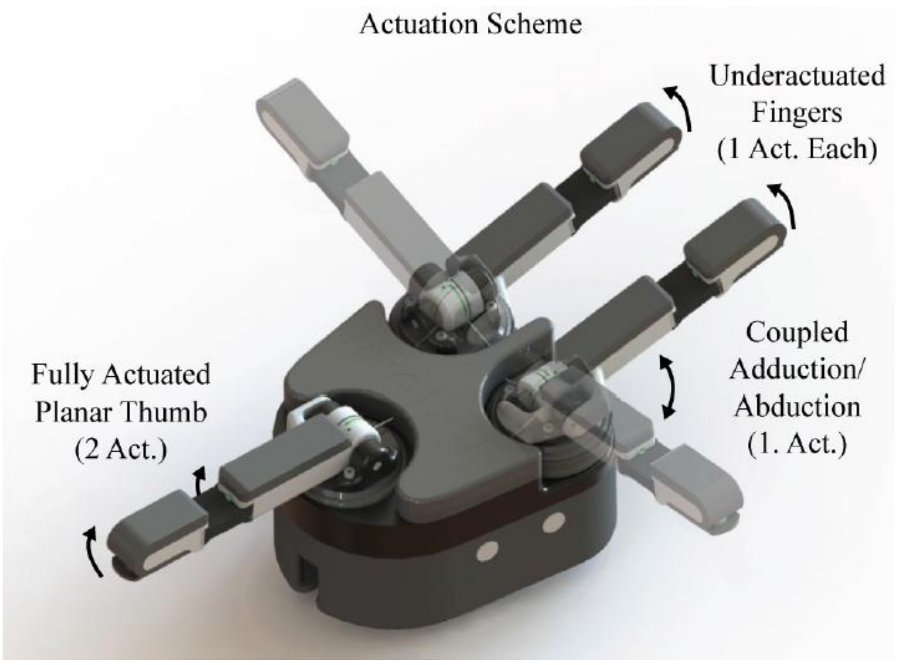

Actuation Scheme
Underactuated Fingers (1 Act. Each)
Coupled Adduction/ Abduction (1. Act.)
Fully Actuated Planar Thumb (2 Act.)

Figure 2-2-2: Actuation scheme of the three fingers robotic hand

Without concern on the fully actuated structure of the finger since 2 actuators are used to control 2 DOFs because the contraction and the relaxing degree of the finger is fully control by the actuators. Instead, we may concern on the underactuated structure fingers. The contraction of underactuated structure fingers of the robotic hand is done by pulling a string that tie on each fingertip. Following diagram shows the basic working principle on contraction of finger of the robotic hand discussed.

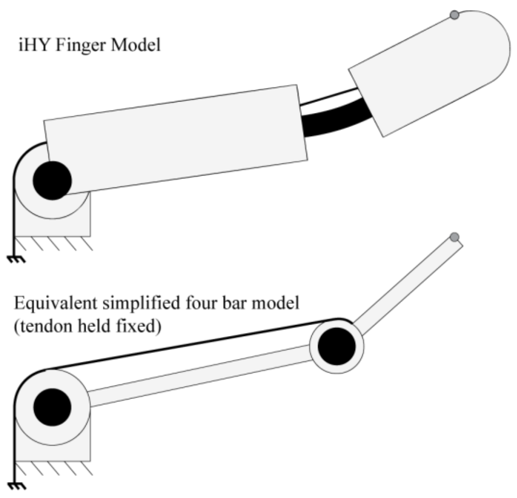


Figure 2-2-3: Basic working principle three fingers robotic hand

To allow fingers relaxing after contraction, elastic and flexure material has been put at the joint of each finger as shown in diagram below.

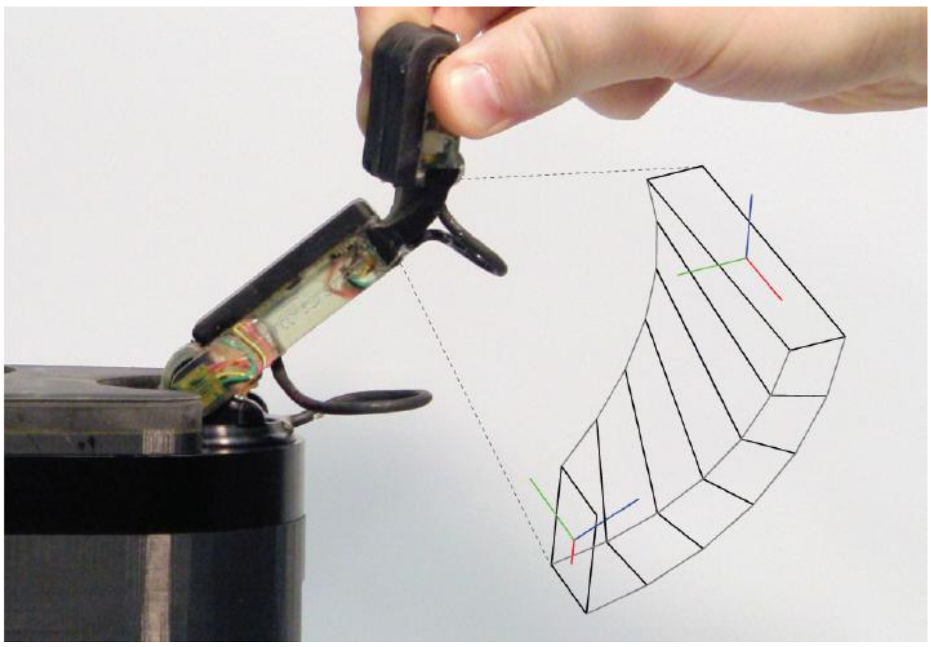


Figure 2-2-4: Elastic material put in each joint

However, this design of robotic hand requires developer to make the hand on their own since the researchers are using 3D printer to print the parts of the fingers then only assembles the parts to become a robotic hand.

Another researcher also comes out with 3 fingers robotic hand [5] but the hand structure is different from the previous research. 3 fingers are place and align together and mounted to a palm. It is a underactuated structure since the whole fingers are construct with only 1 section part with no joint between each section. The actual design view of the robotic hand stated is shown in diagram below.

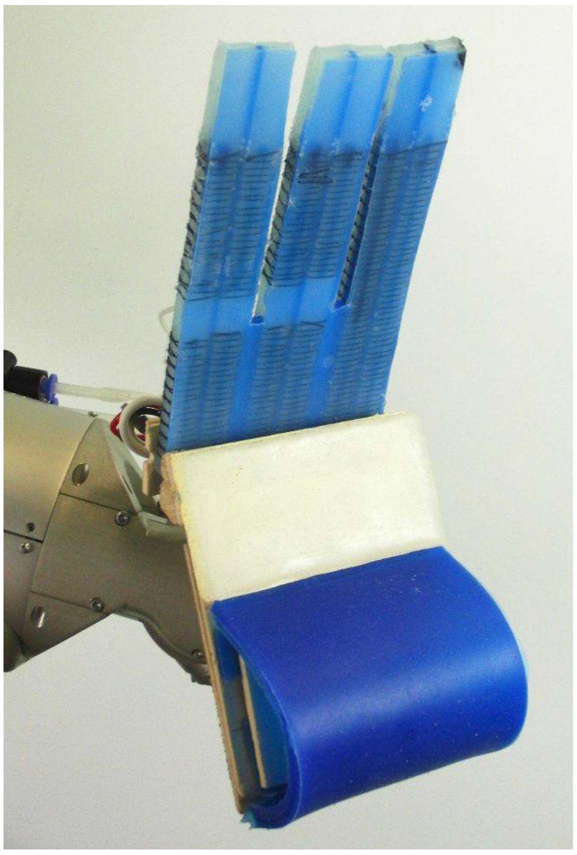


Figure 2-2-5: Actual design view of the three fingers robotic hand

Based on the design, it is greatly for holding an object. Any shape of the object can be perfectly held by the robotic hand. However, there is also weakness for this design. It can hold the object perfectly but it cannot grasp a very small object from a flat surface. For example, when we put a ring on the table, this robotic hand can't actually grasp the ring and hold it because small object may easily stuck under the palm side since the palm is in curve shape.

A research on constructing a robotic hand based on human hand structure [6] has been publishing. This design of robotic hand consists of five fingers with each fingers consists of 3 joints. The diagrams below show the robotic hand structure designed by the researchers and the finger's bone structure which will be used to illustrate the robotic finger structure later.

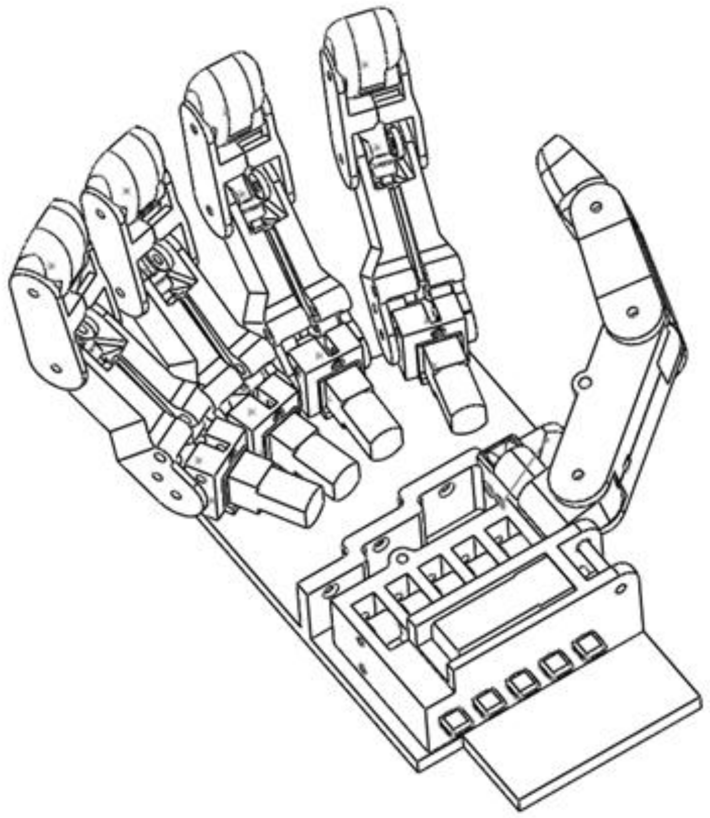


Figure 2-2-6: Five fingers robotic hand structure

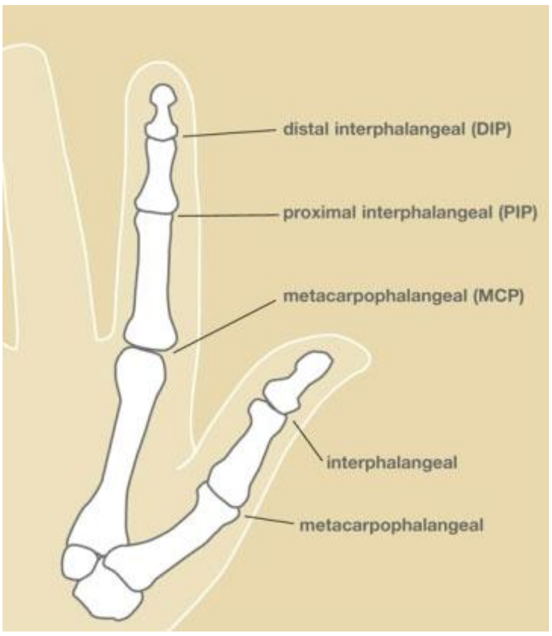

distal interphalangeal (DIP)
proximal interphalangeal (PIP)
metacarpophalangeal (MCP)
interphalangeal
metacarpophalangeal

Figure 2-2-7: Finger's bone structure

There are 10 actuators used by this robotic hand. Each fingers had a combination of underactuated and fully actuated structure. The underactuated structure apply to the first 2 joints, DIP and PIP joints coupled while the fully structure apply to the MCP joints. The following diagram shows the actuation scheme of the robotic hand stated.

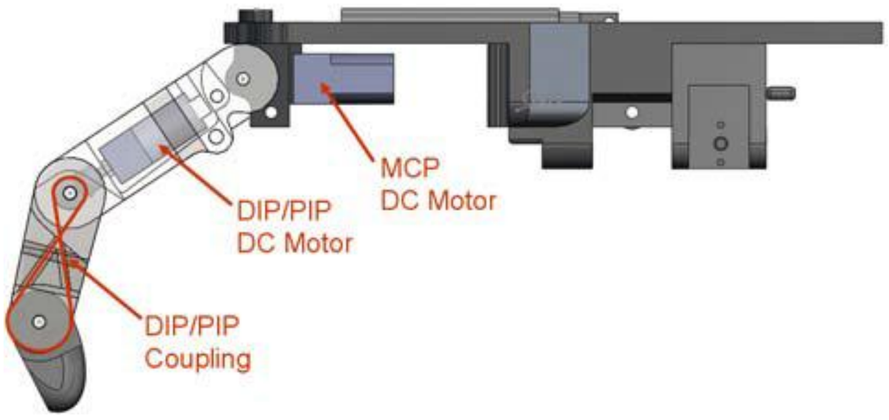


Figure 2-2-8: Five fingers robotic hand's finger actuation scheme

2.3 Robotic Arm

To proceed with robotic arm design, there is some study on the theory need to be carried out first before the development on the structure of the robotic arm. Kinematic study is the most importance instead. Kinematic refers to a study on describing the motion. There are 2 groups divided by kinematic, which are forward kinematic and inverse kinematic. Without concern on the proven of formula and theory, the forward kinematic means to get the coordinate of an end effector from given angles of all the joints included. Meanwhile, inverse kinematic is in opposite. It is define to get angles of all the joints included from a given coordinate of an end effector.

A robotic arm with 4 DOFs has been developed by the researchers [7] which consist of 4 motors. Following diagram shows the general structure of the robotic arm.

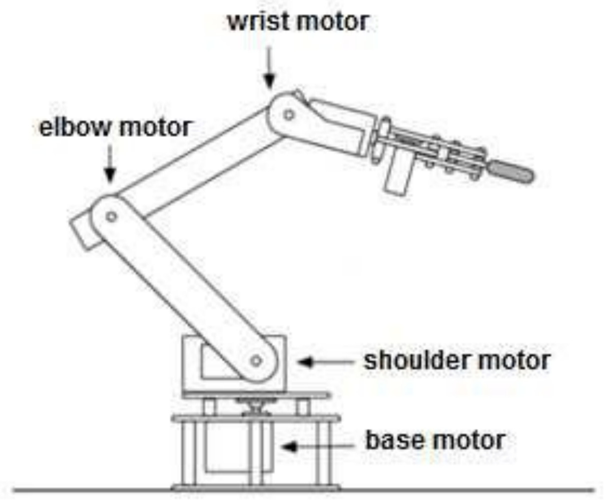


Figure 2-3-1: General structure of the robotic arm

The researchers mainly concern on the inverse kinematic analysis for the robotic arm they design. The researchers able to get the angle of the wrist motor, elbow motor, shoulder motor and the base motor of the robotic arm based on a given coordinate. Following diagram shows the parameter used to define he robotic arm and the mathematical steps define by the researcher using the parameter shown in the diagram to get the respective angles of the motor.

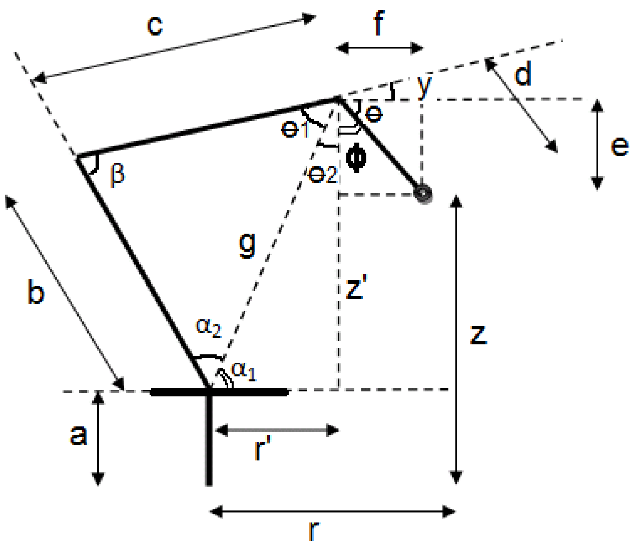


Figure 2-3-2: Angle and parameter of the robotic arm

1) Given (x, y, z) coordinate of end effector.

2) Let base length = a, shoulder length = b, arm length = c, gripper length = d, gripper angle = θ and radial length = r.

3) Base angle = $\tan^{-1}$ (y/x)

4) r = $\sqrt{x^2 + y^2}$, e = d x sin(θ), f = d x cos(θ), r'=r – d x cos(θ), z'=z + d x sin(θ) - a

5) g = $\sqrt{[(r')^2 + (z')^2]}$

6) $\alpha_1 = \cos^{-1}\{[g^2+(r')^2-(z')^2] / (2 \times g \times r')\}$, $\alpha_2 = \cos^{-1}\{[g^2+b^2-c^2] / (2 \times g \times b)\}$

7) Shoulder angle = $\alpha_1 + \alpha_2$

8) $\beta = \cos^{-1}\{[c^2+b^2-g^2] / (2 \times c \times b)\}$

9) Arm angle = 180 – β

10) $\theta_1 = 180 - (\beta + \alpha_2)$, $\theta_2 = 180 - (\alpha_1 + 90)$, ø = 90 - θ

11) Wrist angle = 180 – ($\theta_1 + \theta_2$ + ø)

2.4 Robot Control

There is a research propose using hand gesture to control the robotic vehicle [8]. The robotic car is able to moving forward, reverse, left or right by performing some hand gesture. A 2-axis accelerometer is used to get the x-axis measurement and the y-axis measurement of the human hand and transmit the data to the robotic car via RF. Thus, based on the x-axis measurement and the y-axis measurement get from the accelerometer, researcher decided to use x-axis measurement to perform left and right turn of the robotic car while y-axis measurement to perform forward and reverse of the robotic car. Diagram below shows the full system flow of the program that transforms hand gesture into action of the robotic car.

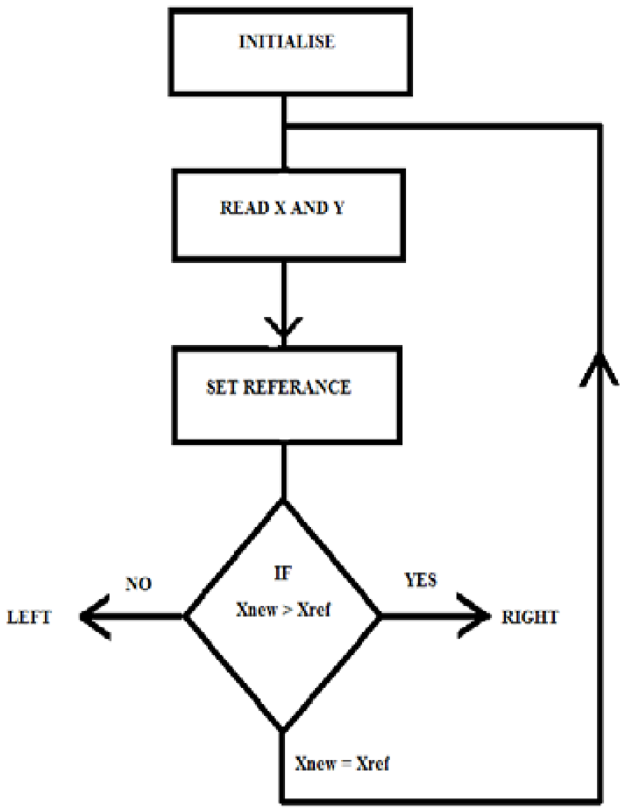


Figure 2-4-1: System flow of the hand gesture control program

Another research also used the hand gesture to control the robot [9], but this time, it is controlling the robotic arm. For the robotic arm part, total of 6 servos is used to construct with 4 servos used to control each joints, 1 servo used to control the rotation of the full robotic arm and 1 servo used to control the gripper. The following diagrams show the mechanical view of the robotic arm assembles in this research.

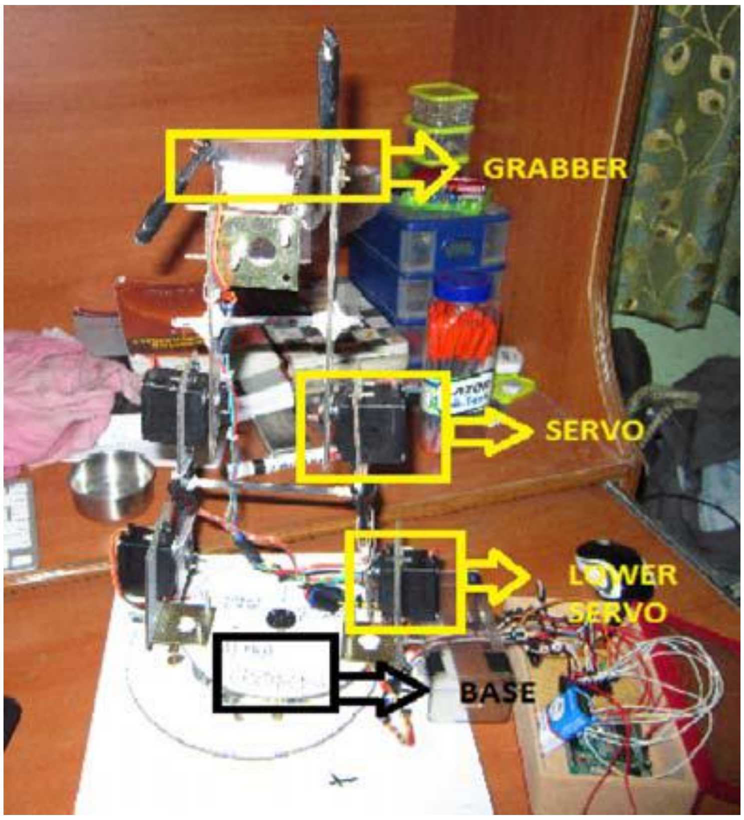


Figure 2-4-2: Robotic arm servo alignment front view

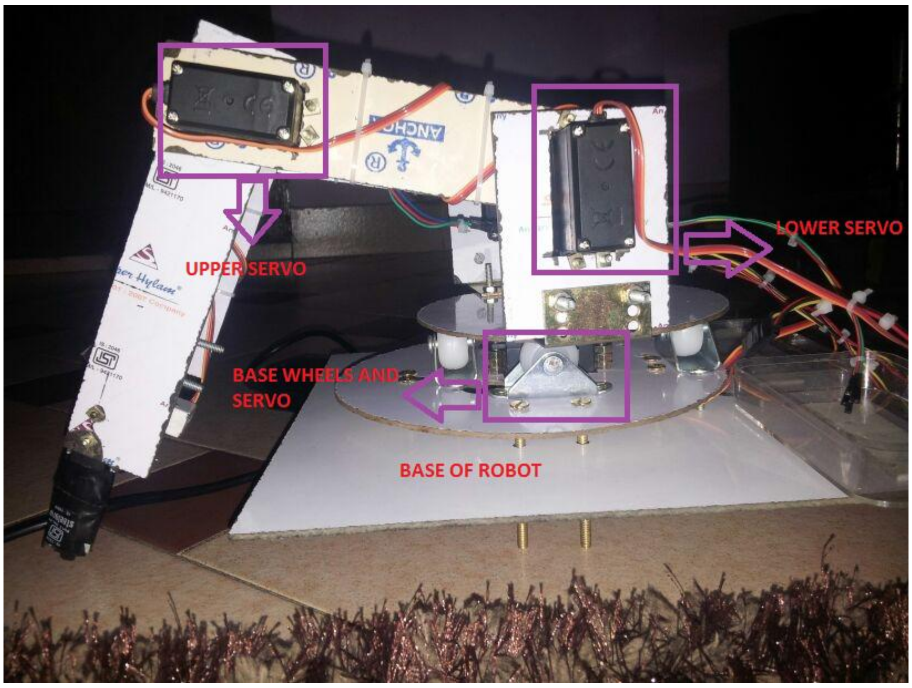


Figure 2-4-3: Robotic arm servo alignment side view

In this research, total 3 sensors used by researcher to control the robotic arm, which are 3-axis accelerometer, 3-axis gyroscope and flexible bend sensor. Accelerometer and gyroscope are used together to control the motion of the robotic arm while the flexible bend sensor is used for controlling the gripper of the robotic arm.

A research of using flexible bend sensor to get the fingers motion [10] has been carrying out. The researchers used the flexible sensor to measure more than 10 DOFs of the hand and had found out the relationship between the joints. Following equations show the relationship between the joints and the range of bending degree of each joint.

1) $\theta_{DIP} = 0.67\ \theta_{PIP}$

2) $0^o \leq \theta_{PIP} \leq (90^o \sim 100^o)$

3) $0^o \leq \theta_{thumb\ IP} \leq 90^o$

4) $0^o \leq \theta_{MCP} \leq 90^o$

Based on the equations shown, researcher able to get the accurate data of the fingers motion and process the data into next level by identifies the hand gesture. Following diagram shows the researcher's application development by using the data get from the flexible bend sensor.

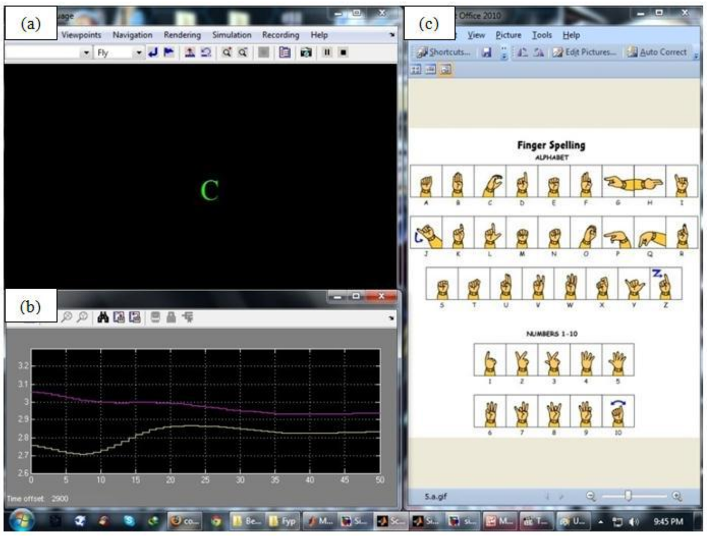


Figure 2-4-4: Result of hand gesture control

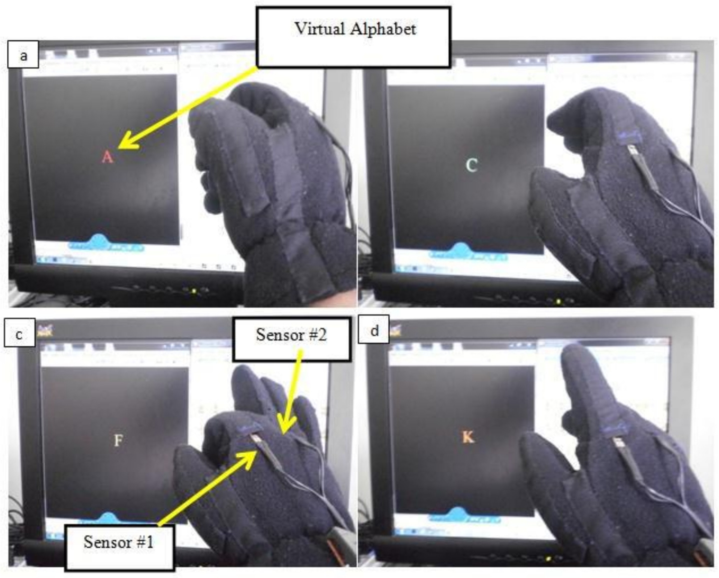


Figure 2-4-5: Result of hand gesture control

2.5 Fire Detection

A research on image processing based fire detection [11] using RGB and YCbCr color space has been carrying out. The algorithm used is based on color detection rather than motion detection and edge detection. Following diagram shows the system flow developed by the researcher.

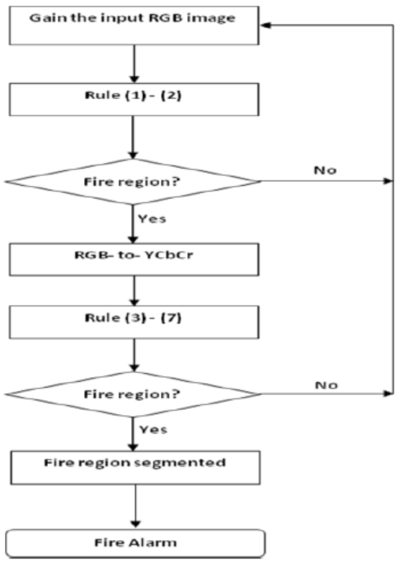


Figure 2-5-1: RGB and YCbCr color space fire detection system flow

For a normal color image, it is in RGB color space with 8-bit value in each layer represent the intensity of the pixel from the range of 0 to 255. For YCrCb color space, Y represent the luminance of the image, Cb represent the chrominance blue while Cr represent the chrominance red of the image. To convert to YCbCr color space, following shows the equation of the respective conversion.

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.2568 & 0.5041 & 0.0979 \\ -0.1482 & -0.2910 & 0.4392 \\ 0.4392 & -0.3678 & -0.0714 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix}$$

The researcher has comes out with a set of rules that can accurately detect the color of the flame. Following shows the rules that define by the researchers.

$$R_1(x,y) = \begin{cases} 1, \text{if } R(x,y) > G(x,y) > B(x,y) \\ 0, \text{otherwise} \end{cases} \quad \text{-------} \ (1)$$

$$R_2(x,y) = \begin{cases} 1, \text{if } (R(x,y) > 190) \cap (G(x,y) > 100) \cap (B(x,y) < 140 \\ 0, \text{otherwise} \end{cases} \quad \text{-------} \ (2)$$

$$R_3(x,y) = \begin{cases} 1, \text{if } Y(x,y) \geq Cb(x,y) \\ 0, \text{otherwise} \end{cases} \quad \text{-----} \ (3)$$

$$R_4(x,y) = \begin{cases} 1, \text{if } Cr(x,y) \geq Cb(x,y) \\ 0, \text{otherwise} \end{cases} \quad \text{------} \ (4)$$

$$R_5(x,y) = \begin{cases} 1, \text{if } (Y(x,y) \geq Y_{mean}(x,y)) \cap (Cb(x,y) \leq Cb_{mean}(x,y)) \\ \quad \cap (Cr(x,y) \geq Cr_{mean}(x,y)) \\ 0, \text{otherwise} \end{cases} \quad \text{--------------} \ (5)$$

$$R_6(x,y) = \begin{cases} 1, \text{if } |Cb(x,y) - Cr(x,y)| \geq Th \\ 0, \text{otherwise} \end{cases} \quad \text{----} \ (6)$$

$$R_7(x,y) = \begin{cases} 1, \text{if } (Cb(x,y) \leq 120) \cap (Cr(x,y) \geq 150) \\ 0, \text{otherwise} \end{cases} \quad \text{---} \ (7)$$

For the rule 5, extra calculations are needed, which are calculating the mean of Y, Cb and also Cb. The formulas for respective calculations are shown in below.

$$Y_{mean}(x,y) = \frac{1}{M \times N} \sum_{x=1}^{M} \sum_{y=1}^{N} Y(x,y)$$

$$Cb_{mean}(x,y) = \frac{1}{M \times N} \sum_{x=1}^{M} \sum_{y=1}^{N} Cb(x,y)$$

$$Cr_{mean}(x,y) = \frac{1}{M \times N} \sum_{x=1}^{M} \sum_{y=1}^{N} Cr(x,y)$$

Note that, all the rules applied are in spatial domain. With a series of rules applied, flame can be accurately extracted out from the original image and researcher concludes that it can achieve 99% of flame detection rate. Diagram below shows the result of the algorithm developed by researcher.

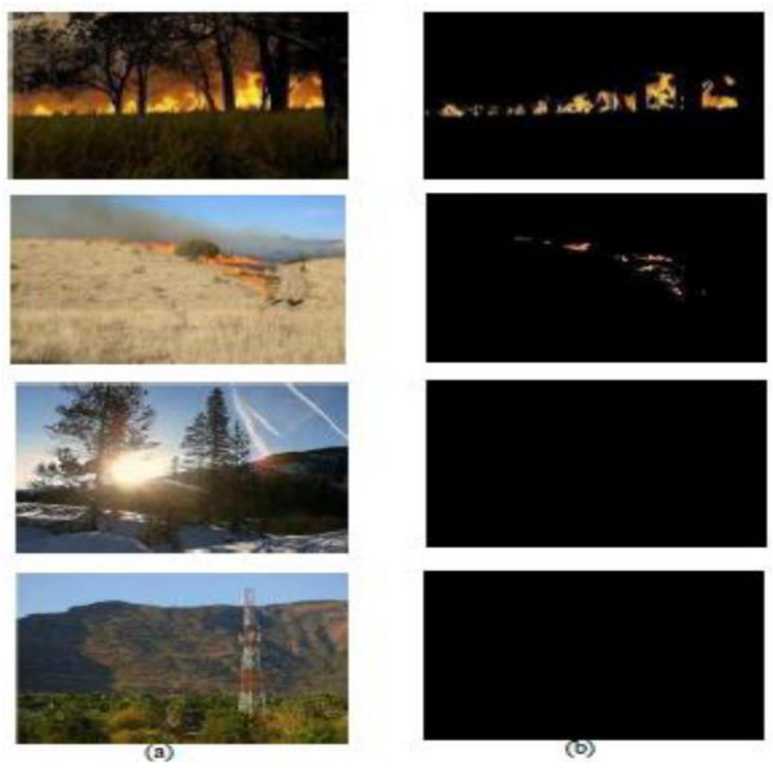


Figure 2-5-2: Output result. a) Original image, b) Output image

However, since there are a lot of rules applied to detect the flame, and thus the complexity of the program and the execution time of the application processing time also increase. In order to overcome the processing time problem, parallel computing would help to increase the performance of the application but the number of CPU cores will be an another issue.

Another researcher using color detection to detect the flame [12] has been carrying out. For color space used, the researcher using HSV and YCbCr color space to detect the flame. HSV color space contain the spatial domain information of the image with H represent the hue of the image, S represent the saturation of the image and V represent the intensity of the image. Following equation shows the rules define by researcher.

$$f(x,y) = \begin{cases} 1 & \begin{aligned} &\mathit{if}\,((0.02 < \mathrm{H(x,y)} < 0.30) \text{ and} \\ &\quad (0.20 < \mathrm{S(x,y)} < 1.00) \text{ and} \\ &\quad (0.98 < \mathrm{V(x,y)} < 1.00) \text{ and} \\ &\quad (\mathrm{Y(x,y)} >= \mathrm{Ymean}) \text{ and} \\ &\quad (\mathrm{Cb(x,y)} <= \mathrm{Cb\ mean}) \text{ and} \\ &\quad (\mathrm{Cr(x,y)} >= \mathrm{Cr\ mean})) \end{aligned} \\ 0 & \mathit{otherwise} \end{cases}$$

After developed the application of the algorithm, researcher conclude that the final result of this algorithm achieve 90.73% accuracy on fire image and 98.13% on non-fire image. However, in the sense of accuracy, it is rather low compare with the previous research that discussed with accuracy rate achieve 99%.

Chapter 3 Hardware Development

3.1 Microcontroller and development board

In this project, dsPIC30F4013 microcontroller is used as the controlling and processing unit of the robot. This microcontroller is manufactured by Microchip Inc. It is a RISC ISA design CPU with a total of 84 instructions and it is a 16-bit microcontroller with each instruction of 24-bit wide. Following table shows the basic specification of the dsPIC30F4013 microcontroller.

Device	dsPIC30F4013
Pins	40
Operating Voltage	2.5V ~ 5.5V
Program Memory	24 kbytes (16k instructions)
Timer	5
PWM	4
ADC channel / resolution	13 / 12-bit
Communication interface	2 x UART, I2C, SPI

Table 3-1-1: Specification of dsPIC30F4013 microcontroller

For ease of development, the SKds40A development board is used. This development board is designed and manufactured by Cytron Technologies Sdn Bhd and it is support for dsPIC30F3011, dsPIC30F4011, dsPIC30F3014 and dsPIC30F4013 microcontroller. Following diagram shows the board layout of the SKds40A development board and the table shows the layout description of the development board.

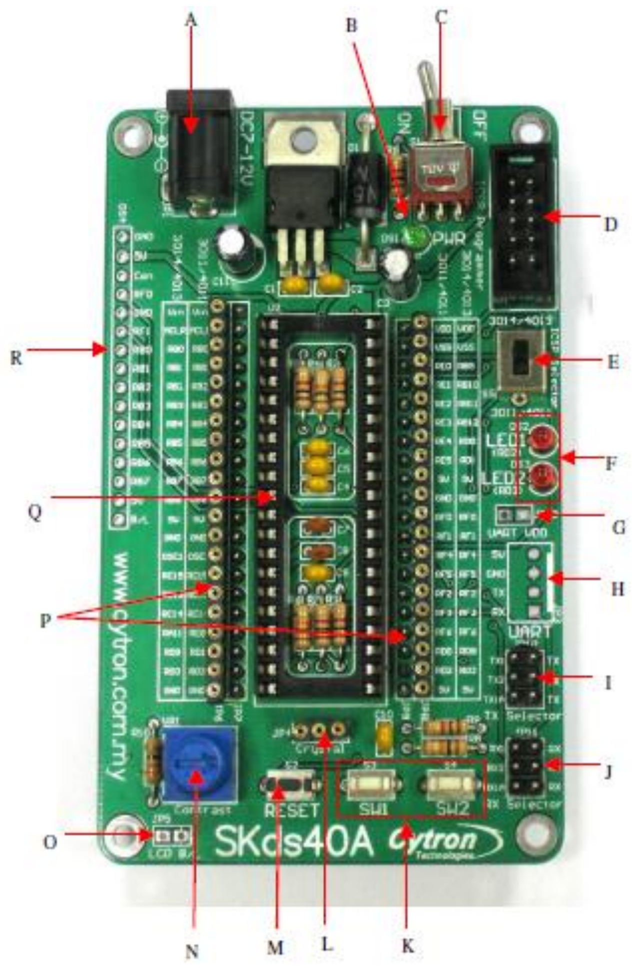


Figure 3-1-1: Layout view of the SKds40A development board

Label	Function
A	DC power adaptor socket
B	Power indicator LED
C	Toggle Switch for power supply
D	Connector for UIC00B Programmer
E	ICSP Selector
F	Programmable LEDs
G	UART Power
H	UART Connector
I	Tx Selector

J	Rx Selector
K	Programmable Push Button
L	Turn pin for external crystal oscillator
M	Reset button
N	LCD contrast
O	JP5 for LCD Backlight
P	Header pin and turn pin
Q	40 pin IC socket for microcontroller
R	Pads for 2x16 Parallel LCD

Table 3-1-2: SKds40A development board layout description

3.2 Robotic car

In this project, robotic car consists of 4 wheels is choose as the moving based for the mobile robot. Each wheel is control by a dc motor to rotate clockwise or anti-clockwise. The basic circuitry for each dc motor's rotating direction control is based on H-bridge method in which is shown below.

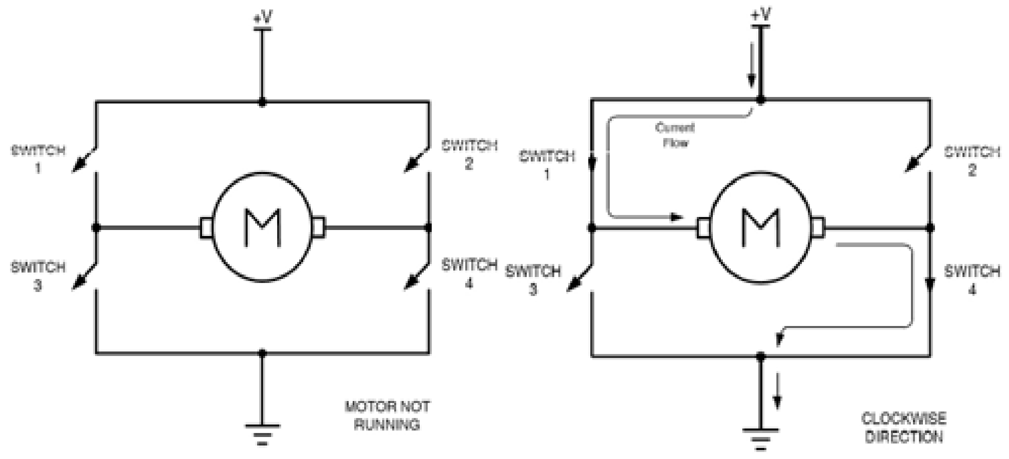


Figure 3-2-1: H-bridge working principle

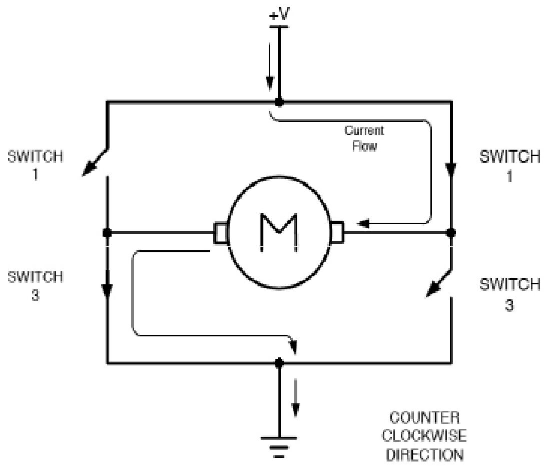


Figure 3-2-2: H-bridge working principle

With refers to diagram above, when all 4 switches of the H-bridge are open or close, the dc motor is not running and hence the robotic car will not moving forward or backward. When switch 1 and 4 is closed, the dc motor rotates in clockwise direction. When switch 2 and 3 is closed, the dc motor rotates in counter clockwise.

For this project, a L298N motor driver module is used. This module consists of dual H-bridge circuitry integrated inside chip and 5V regulated output voltage with maximum output current up to 2A. Using 2 H-bridge circuit instead of 4 to control the robotic car is more than enough since it can significant reduce by half the GPIO pin of the microcontroller from 12 pins consists of 8 pins to 4 H-bridge and 4 pins for PWM control to 6 pins consists of 4 pins to 2 H-bridge and 2 pins for PWM control. The PWM control will be discuss in section 4.2. Besides that, from the mechanical view, the front wheel and rear wheel of either left side or right side, when moving towards a direction, both sides of dc motor are exactly moving for the same direction. Thus, for 4 wheels robotic car, instead using 4 H-bridge circuit to control, it is better to implement it with 2 H-bridge circuit with each H-bridge control for each side, either left or right. Therefore, L298N motor driver module is very suitable for the robotic car of this project.

Next, with implementing H-bridge for each side either left of right, the robotic car can now moving forwards or backwards with left side motor rotate clockwise and right side motor rotate anti-clockwise or vice versa which means left and right side rotating direction always counter with each other. Next, in order for the robotic car to turn either left or right, following figures show the basic physical mechanism that allow the robotic car to do so.

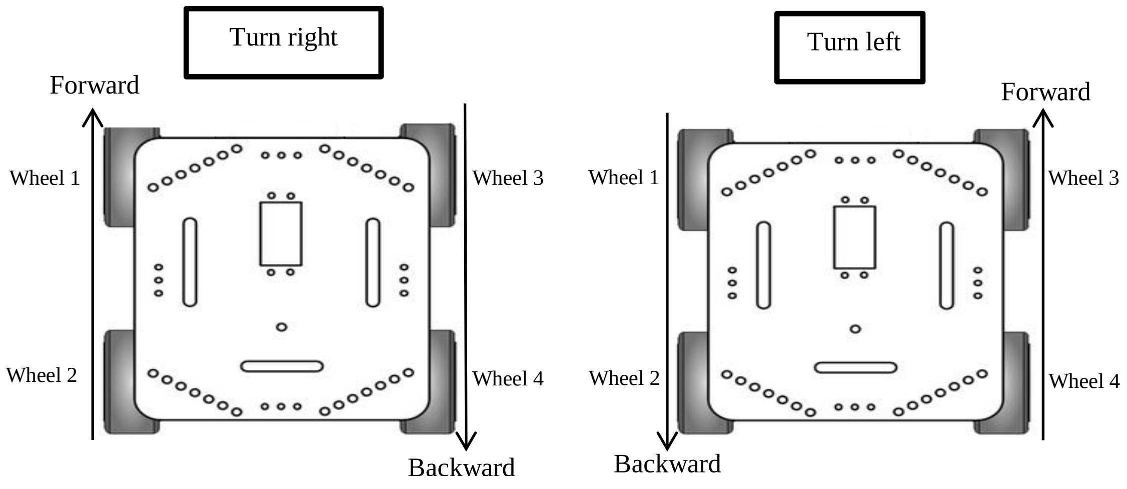


Figure 3-2-3: Robotic car working direction

From the above diagram, to turn right, wheel 1 and wheel 2 dc motor are selected to rotate forward meanwhile wheel 3 and wheel 4 dc motor rotate backward. Whereas to turn left, wheel 3 and wheel 4 dc motor are selected to rotate forward and wheel 1 and wheel 2 dc motor rotate backward.

In mechanical view, the robotic car is now able to move in 4 directions, i.e. forward, reverse, left and right. In order to make the robotic car more flexible to control, it should be design with 8 directions moving, i.e. forward, reverse, left, right, forward-left, forward-right, reverse-left and reverse-right. The existing mechanical part of the robotic car is completely satisfied for all 8 directions and diagram below shows how the basic concept for the robotic car to move to all 8 directions.

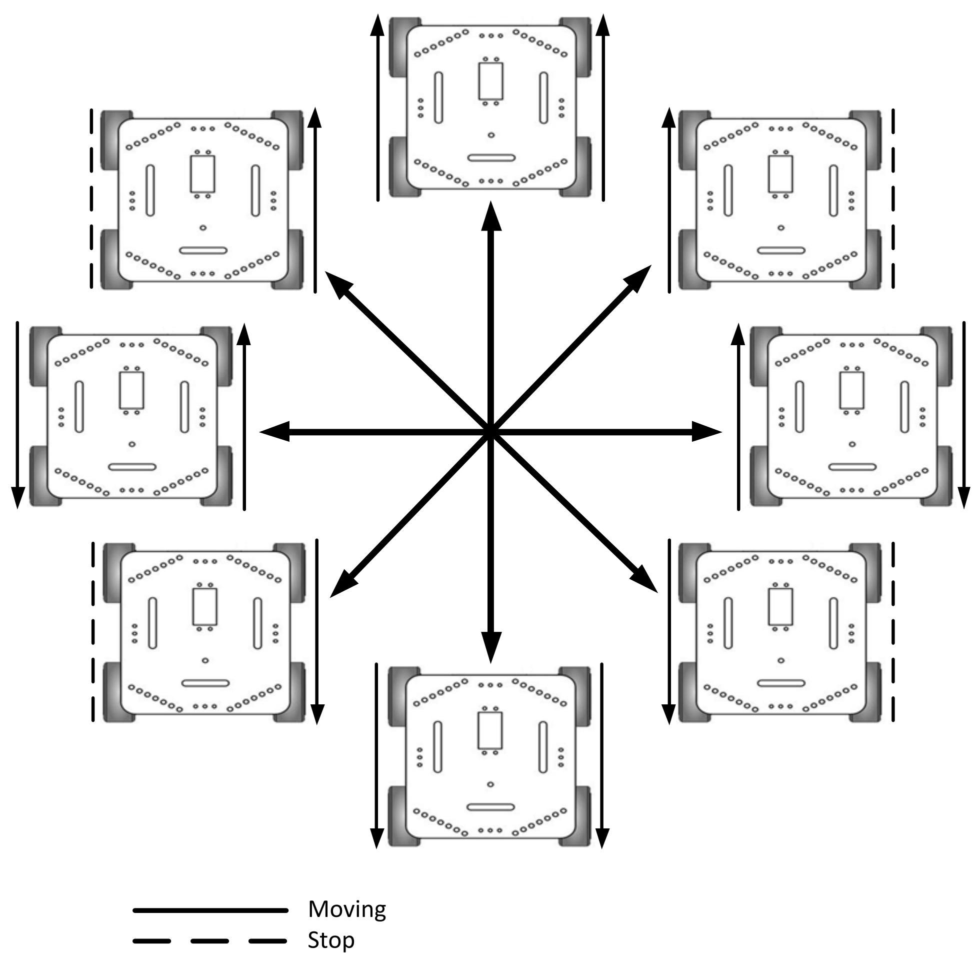


Figure 3-2-4: Robotic car 8-direction moving

Since hardware part had satisfy the concept of 8 direction moving, the implementation remaining can only be done by using software control in which will be discussed in section 4.3. With the concept being discussed in this section, following diagram show the circuit connection for the robotic car with the microcontroller.

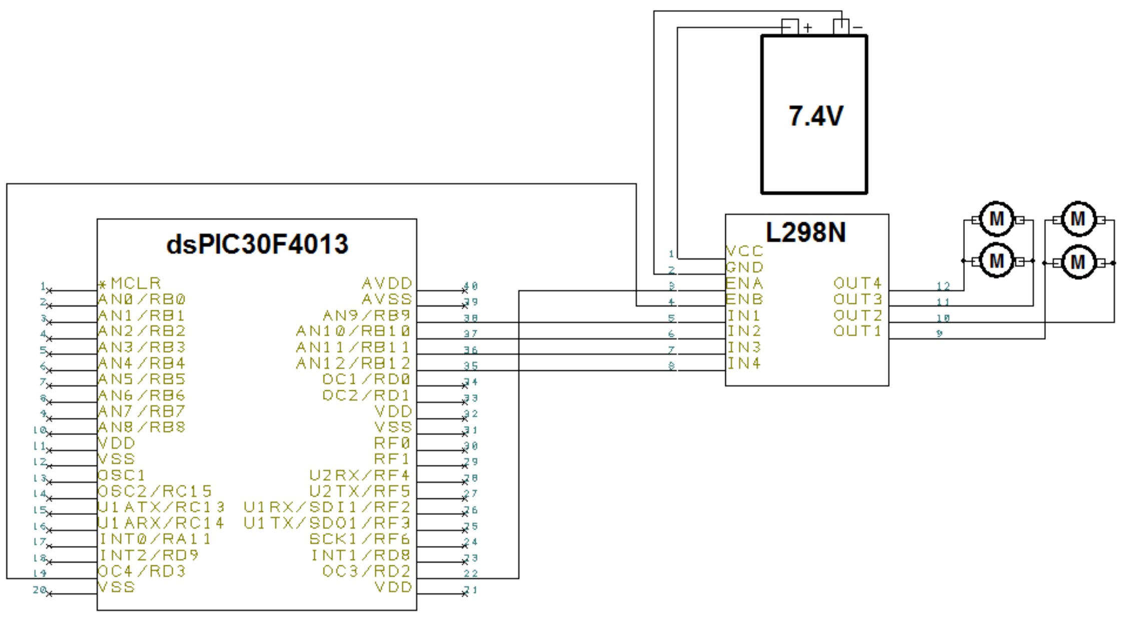


Figure 3-2-5: Schematic view of robotic car

From the circuit diagram, we can conclude that the robotic car needs 2 PWM signal and 4 GPIO in order to fully functional as discussed.

3.3 Robotic hand

One of the interesting parts of this project is that it is equipped with one robotic hand. The robotic hand in this project is human-like hand, in which consists of five fingers and can perform action almost like a normal human's hand. According to our analysis, using human hand-like robotic hand takes advantage in holding or grasps the object more tightly and perfect fit compare to other pattern of robotic hand. The general structure of each robotic hand's finger is as shown in diagrams below.

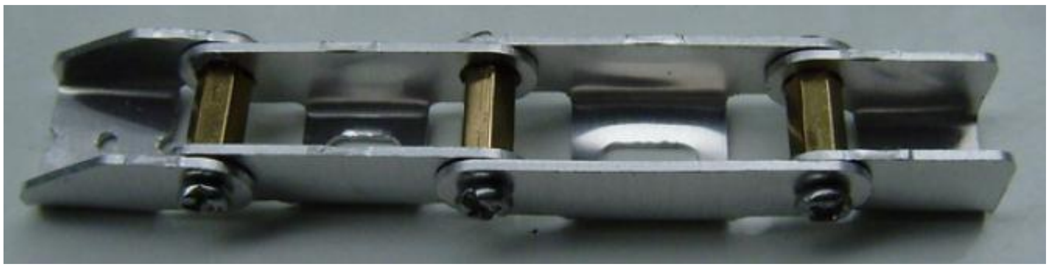


Figure 3-3-1: Structure of robotic hand's finger bottom view

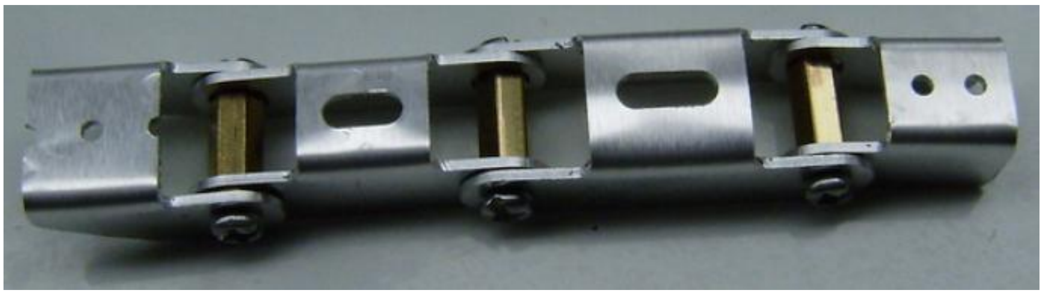


Figure 3-3-2: Structure of robotic hand's finger top view

From the diagrams above, the robotic fingers are made up of metal, and thus it is more durable to use and more strength when hold an object. For normal human fingers, fingers other than thumb have 3 joints, i.e. distal interphalangeal joint (DIP), proximal interphalangeal joint (PIP), metacarpophalangeal (MCP). The thumb has only 2 joints, which are interphalangeal joint (IP) and metacarpophalangeal (MCP). Following diagrams show the human fingers bone structure.

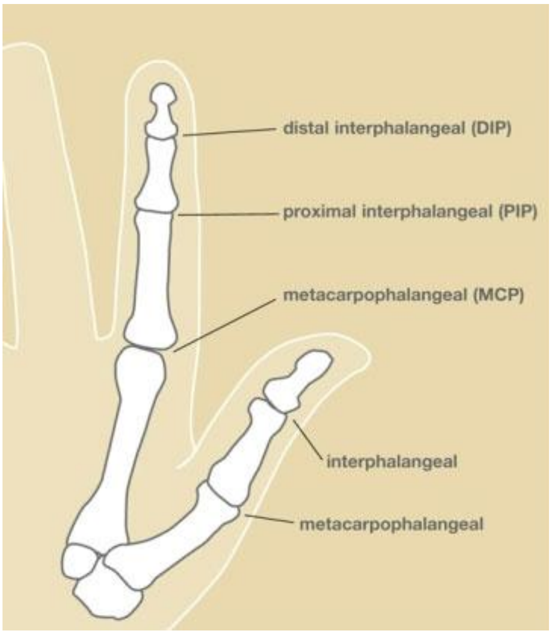

distal interphalangeal (DIP)
proximal interphalangeal (PIP)
metacarpophalangeal (MCP)
interphalangeal
metacarpophalangeal

Figure 3-3-3: Bone structure of thumb

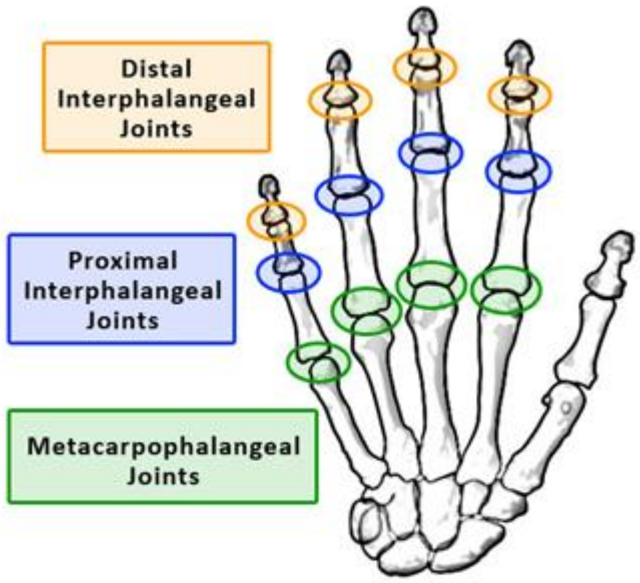

Distal Interphalangeal Joints
Proximal Interphalangeal Joints
Metacarpophalangeal Joints

Figure 3-3-4: Bone structure of other fingers

With each joint represent as one bending section or degree of freedom, all the metal fingers except thumb can be bending into 3 sections while the thumb can only bend into 2 sections, which is similar as the normal human fingers. However, for the case of metal thumb finger, only 2 bending section limit the thumb to hold or even grasp for an object. Thus, it is purposely to design the metal thumb with 3 bending section. Following diagram shows the full hand structure by combine all 5 metal fingers and connect it with a metal palm.

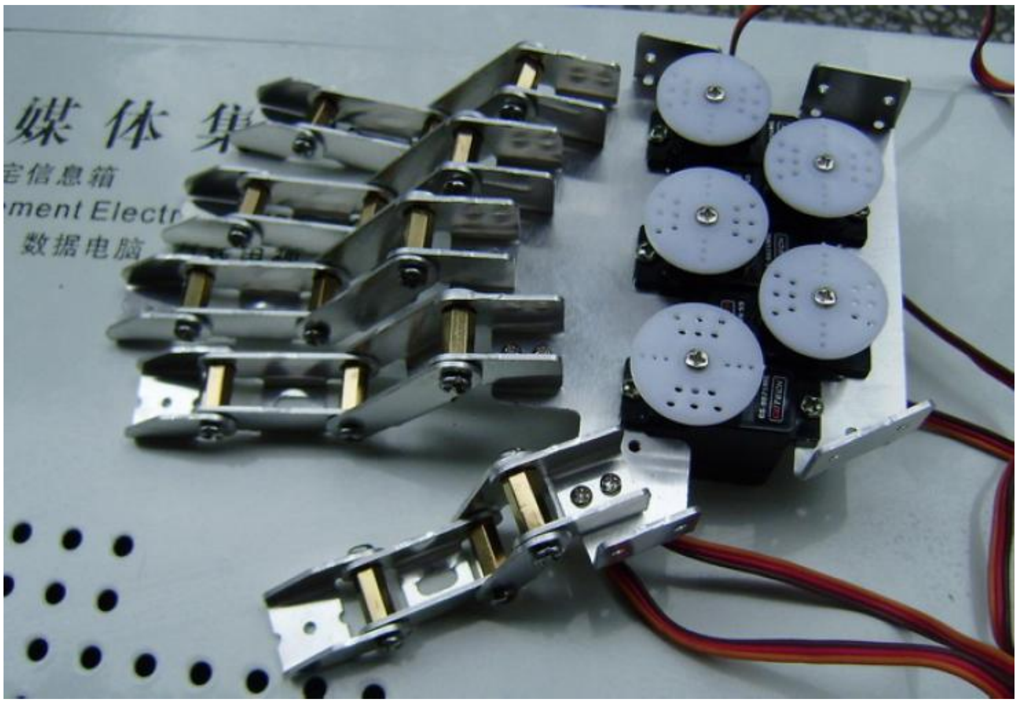


Figure 3-3-5: Robotic hand assembly view

Another thing that we can investigate from the diagram above is that there are 5 servos mount to the metal palm. The main usages of the servos are to control the bending degree of the fingers. The more the servo turns from the original position, the tighter the finger is. Initially, the fingers control by servo is design such that a cable tie’s top end part is tie to the fingertips while another end is tie on the servo.

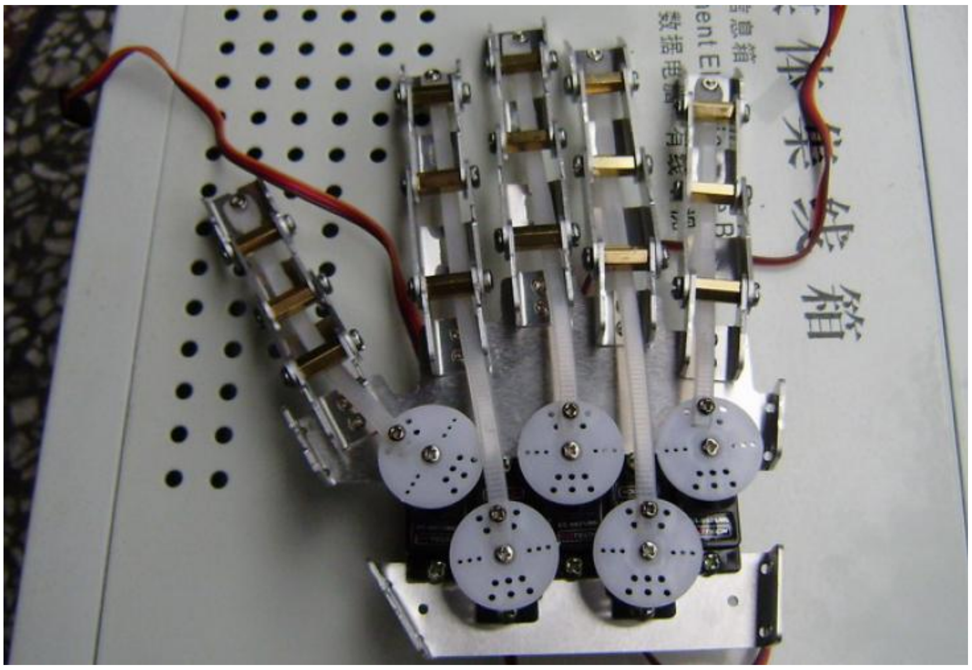


Figure 3-3-6: Robotic hand's fingers initial approach

The main problem for this design is that when all the fingers are bending and wanted to grasp an object, it is less strength when holding the object. Furthermore, the maximum load for a robotic hand is reduce, i.e. each servo is able to support a maximum torque of 2kg/cm and that value will be reduce if using the initial design.

Thus, the second approach has been comes out. For our second approach, we had make the cable tie act as a spring for the robotic finger since the cable tie is elastic and adding fishing string as to pull by servo motor to control the bending degree in which it is for finger's contracting purpose. The advantage of this design is that when grasps or hold an object, it is much stronger than the initial design. The robotic fingers bending more narrow than before and is able to fully use the maximum torque of the servo. The fishing string used is able to hold up to 25 lbs. and thus, it is able strong hold the position for the robotic fingers. Following diagram shows the graphical view of the second approach.

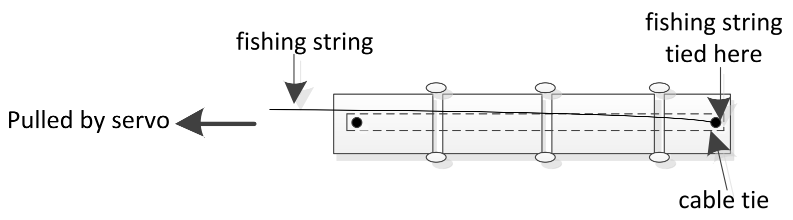


Figure 3-3-7: Robotic hand's fingers second approach

However, there is a disadvantage for this design. The advantage of this design benefits for fingers contracting but in contrast, it is weak when the fingers flip back to relax position. Since the cable tie is used as the spring for the robotic finger to flip back to original position, it is rather depends on the elasticity of the cable tie. When it became frequently, the cable tie's elastic performance will go weaker and thus it is unable to perform finger relaxing well.

Therefore, the third approach which is the final approach comes out. The third approach is success by adding another fishing string to the back of the cable tie. It means that there exist 4 layers for the robotic fingers which are fishing string, robotic finger, cable tie and fishing string. The graphical description of the design of our third approaches is shown in the next diagram.

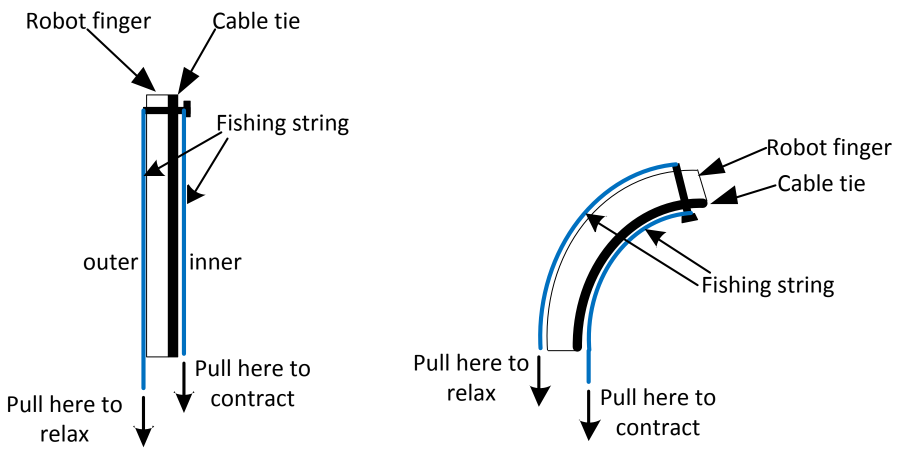


Figure 3-3-8: Robotic hand's fingers final approach

In order to perform relaxing and contracting of the robotic fingers well, another end of both fishing strings that will tie on the servo motor are tie in opposite direction. It means that when first fishing string is pull toward 180 degree of the servo, the second fishing string will be in 0 degree of the servo motor or vice versa in which both strings are always different for 180 degree.

With realizing all the mechanism into the robotic hand, the remaining part is that the software control part which is to program the PWM in order to control the servo motor to rotate and pulled the fishing strings tie on each fingers so that can bend to wanted degree.

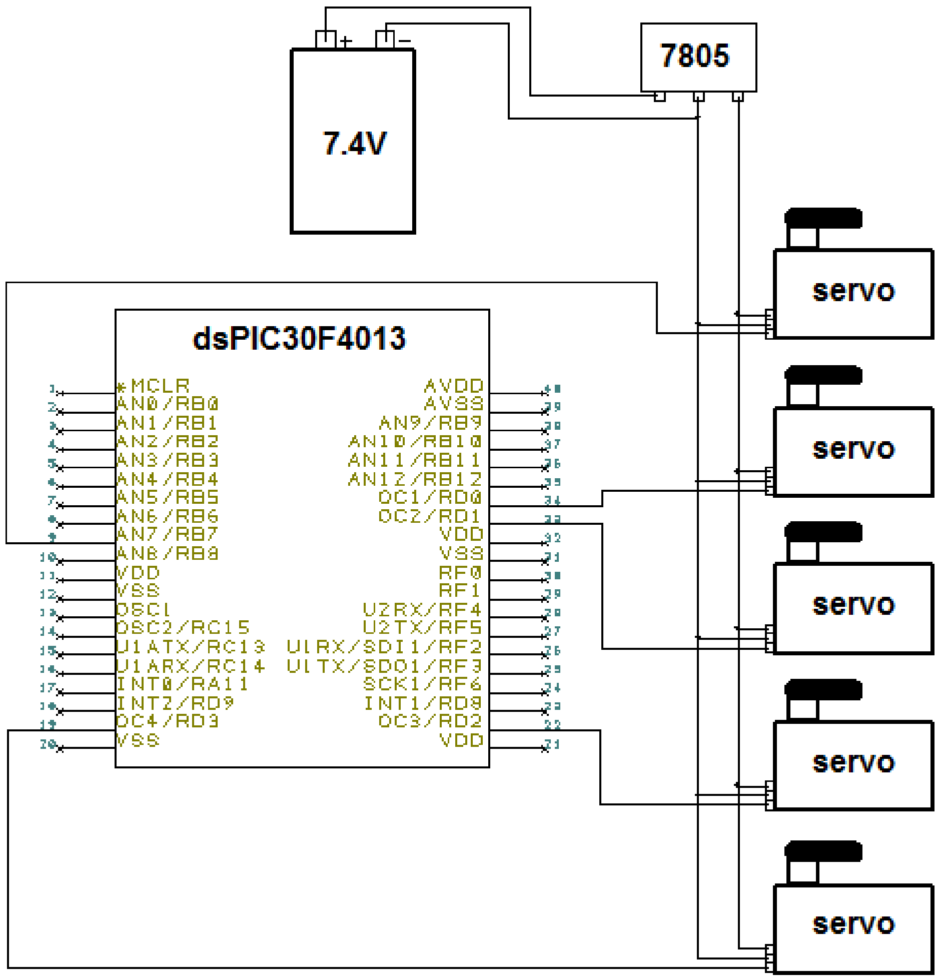


Figure 3-3-9: Schematic view of robotic car robotic hand

3.4 Robotic arm

With the complete robotic hand constructed, next is to construct a robotic arm as to use it to connect with robotic hand and attach the whole part to the robotic car unit. By connecting robotic hand and arm, the final product may perform some action like a human hand and arm action, for example, after grasp an object, robotic hand is lift up by robotic arm. Following diagram shows the working mechanism of the robotic arm.

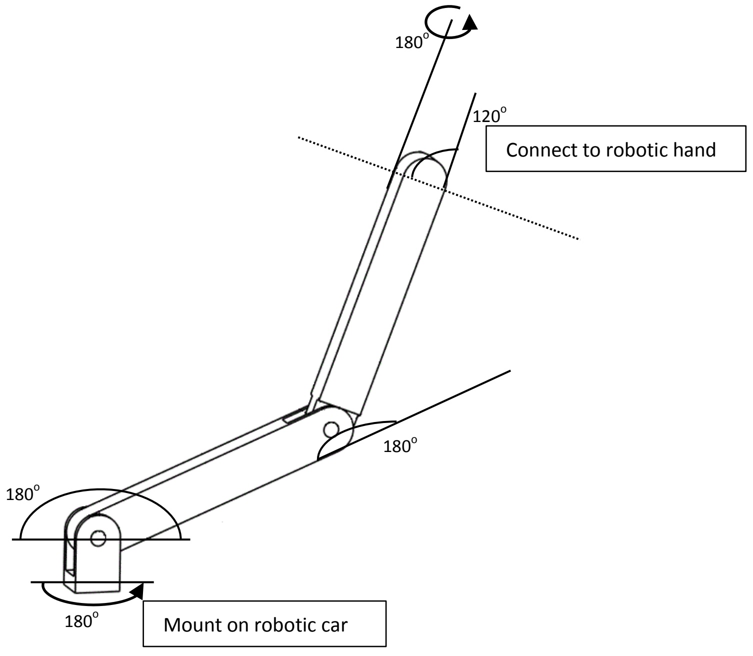


Figure 3-4-1: Working mechanism of the robotic arm

From the diagram above, 5 degree of freedom (DOF) implemented for the robotic arm and thus total 5 servos are used to construct the robotic arm. The material used to construct the robotic arm is acrylic board since it is strong, light and cheap in price, which is the most suitable material to build a robotic arm.

To control the robotic arm, the next thing to do is connect the robotic arm to the microcontroller. The action of the robotic arm is defined by the degree of rotation of the servo motor. In order to control the degree of rotation of the servo, PWM signal needs to be generated by the microcontroller. Apart from this, besides using the PWM peripherals of the microcontroller, using the timer of the microcontroller and generate out the PWM signal through GPIO is also attainable. From previous section, since dsPIC30F4013 has 5 timers integrated inside chip, by assuming the TIMER1 of the microcontroller is used by system and TIMER2 and TIMER3 are used by PWM module, the robotic car has used 2 PWM output to control the movement of the car and left 2 more PWM peripherals and 2 timers to use while the robotic hand has used 4 PWM module and TIMER4 to control the servo motor of the robotic hand and only left 1 timer to use. To fully utilize the resource instead adding another more microcontroller chip, it is exactly perfect to fit 4 of the servo motors of the robotic arm to the robotic car's microcontroller while another 1 to the robotic hand's microcontroller. The circuit connection will be as following diagram.

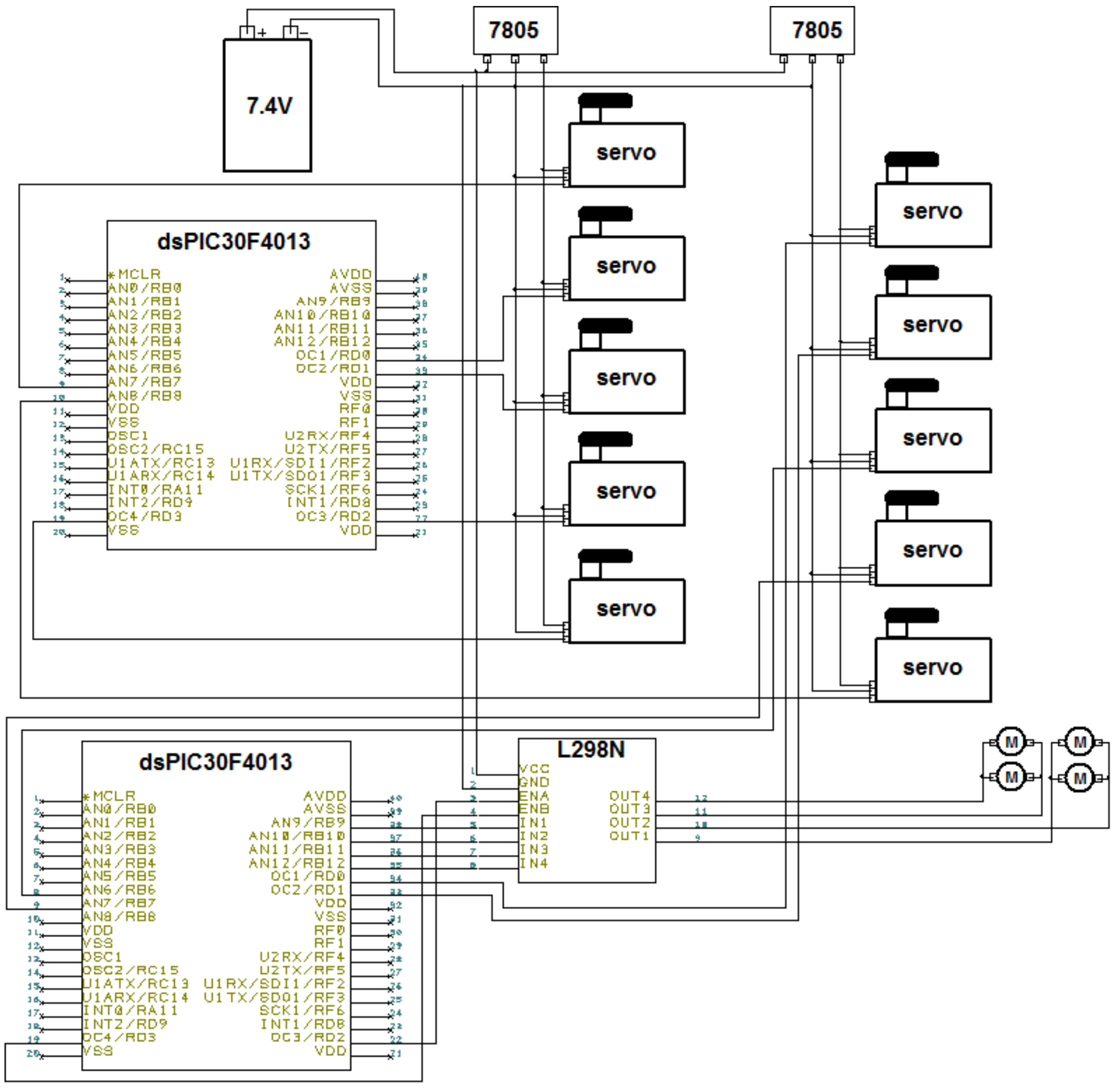


Figure 3-4-2: Schematic view of robotic car robotic arm

3.5 Sensor glove

To control the robotic hand accurately, using a sensor glove is always a better choice rather controlling using joystick. There exists many combination of action of the robotic hand and thus it is rather hard to cover all the combination by using joystick controlling. Using sensor glove can helps to decrease the effort in terms of identify the hard coded hand gesture for the robotic hand to perform. To construct the sensor glove, the flexible bend sensor is used and diagram below shows how the flexible bend sensors are being placed inside a glove.

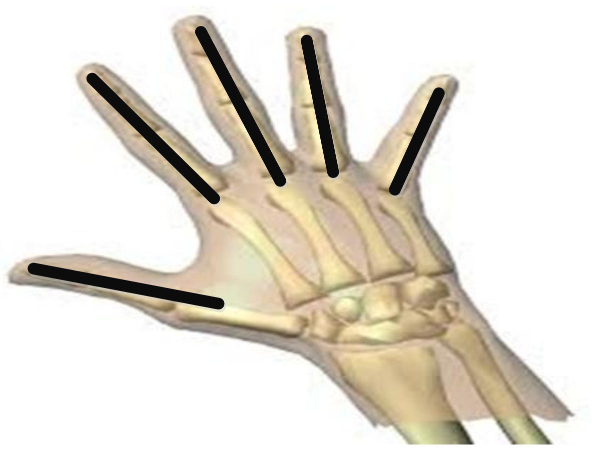


Figure 3-5-1: Placing of flexible sensor to hand glove

Next, in order to get the sensor value of the flexible bend sensor, study of working principle of the flexible bend sensor need to be carry out and following diagram shows how actually the flexible bend sensor is working.

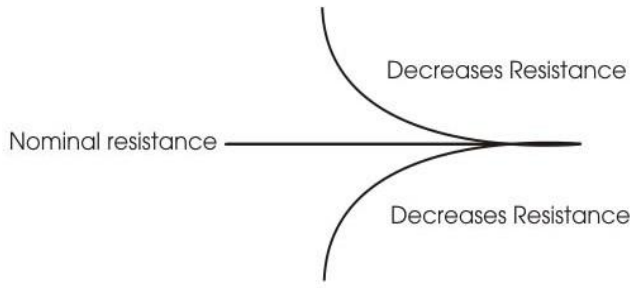


Figure 3-5-2: Flexible bend sensor working principle

With refers to the previous diagram, one thing can be conclude is that the flexible sensor can be act as a potentiometer. Whenever any tuning, the resistance value will be affected. Next, is to get bending degree of a normal human hand's fingers and convert it into electrical signal. To get the resistance value of the flexible sensors, it must be converting into voltage value since ADC module of the microcontroller can only read the voltage of a circuit and thus voltage divider circuit is applied. Following circuit diagram shows the voltage divider circuit connection of the flexible bend sensor.

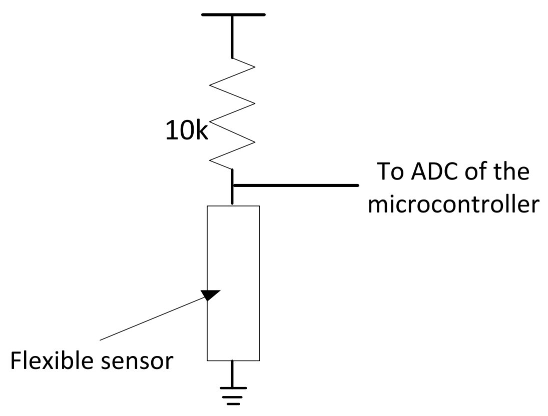


Figure 3-5-3: Voltage divider circuit connections

With using the voltage divider circuit as above, we may get the accurate value of resistance when bending to certain degree and next step to do is to decode the value get and send it to robotic hand's microcontroller to output signal to the servo that attach to the robotic hand so that to perform the action as perform by the human hand when do hand gesture using hand glove. In order to complete the controlling system, the sensor glove is connected to the microcontroller and following diagram shows the connection of both components.

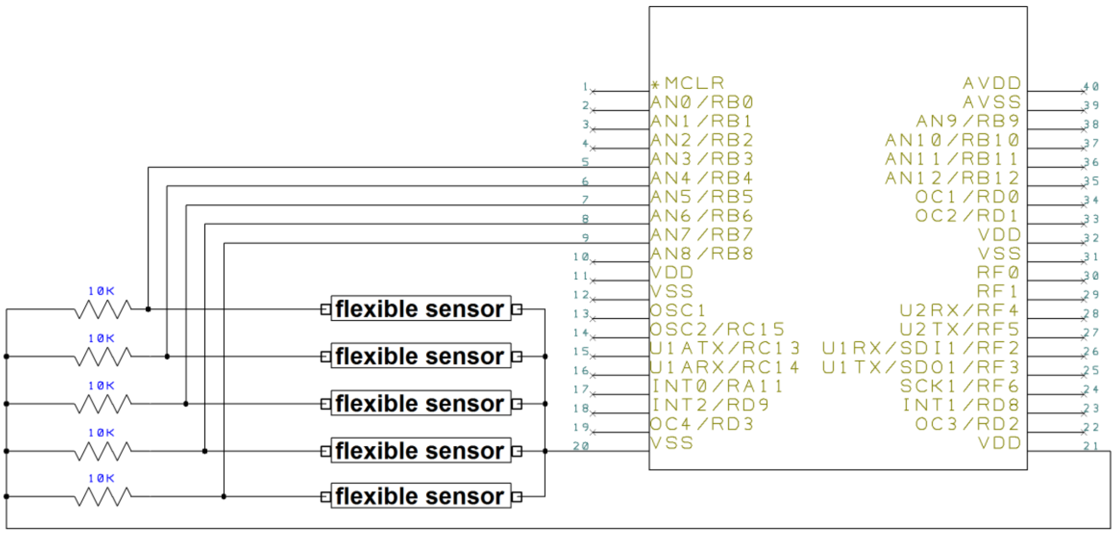


Figure 3-5-4: Schematic view of sensor glove

3.6 Joystick controller

To control the motion of the robotic car and robotic arm, the joystick controller is used. The joystick shield used originally is design for Arduino UNO. It is mainly plug-and-play for Arduino UNO and no extra circuit needed. For our case, since the development board and microcontroller are SKds40A and dsPIC30F4013 respectively instead of Arduino UNO with ATmega328P microcontroller inside, no plug-and-play for the development board of this project and extra circuitry interfacing is needed. First thing to do is to identify the original interfacing circuit between Arduino UNO and joystick shield, that means to find out the pin allocation of the joystick shield and identify the function of each pin. Diagram below shows the joystick shield module and the pin diagram of Arduino UNO interface with joystick shield.

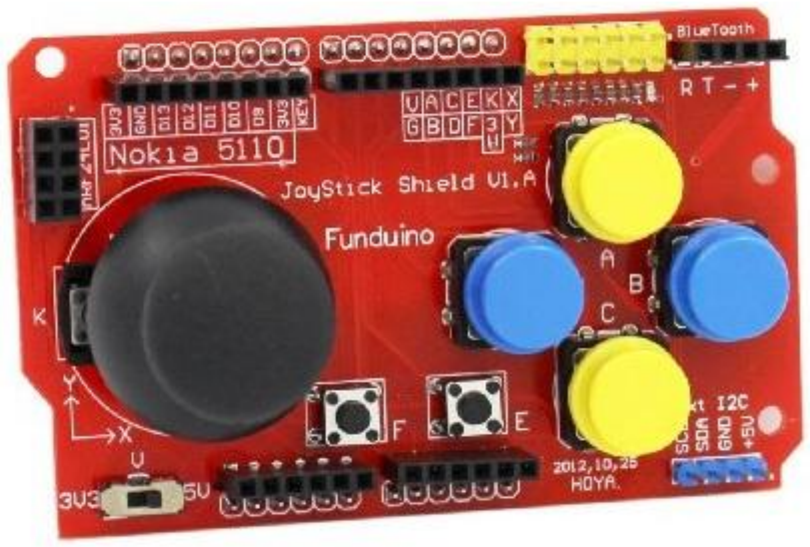


Figure 3-6-1: Actual view of Arduino joystick shield module

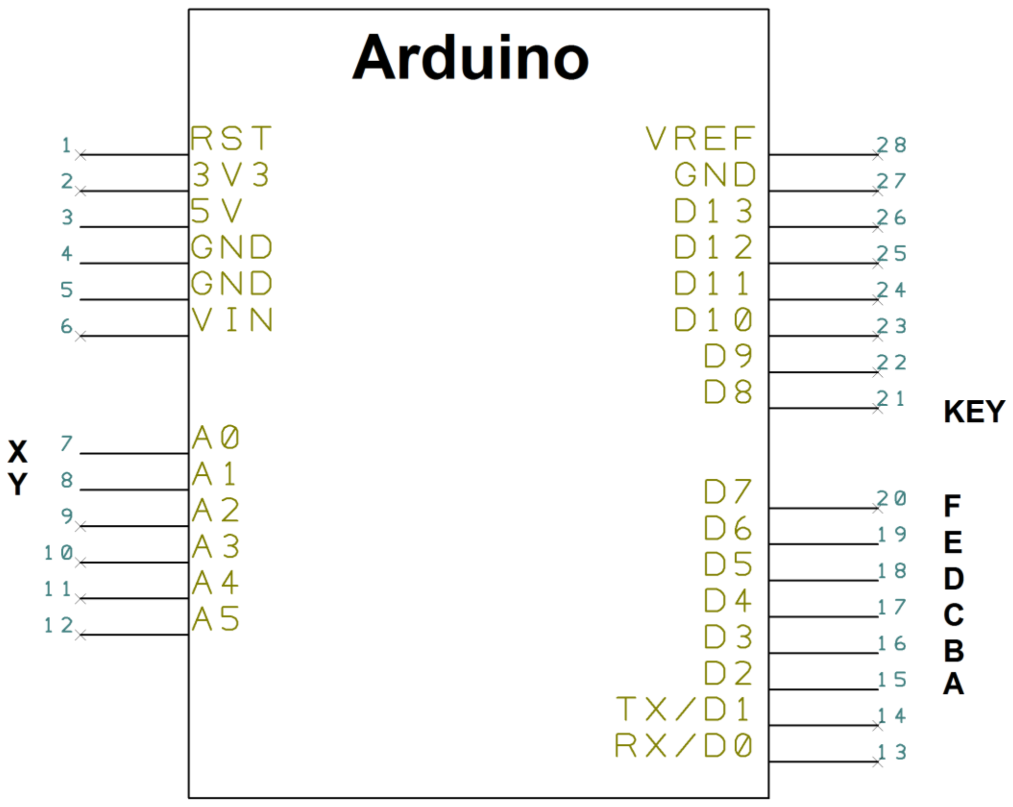


Figure 3-6-2: Pin diagram of Arduino UNO

The table below describes the pin function of the joystick shield module.

Pin	Pin Function
X	x-axis of the analog joystick
Y	y-axis of the analog joystick
A	Assert high: when button A is released Assert low: when button A is pressed
B	Assert high: when button B is released Assert low: when button B is pressed
C	Assert high: when button C is released Assert low: when button C is pressed
D	Assert high: when button D is released Assert low: when button D is pressed
E	Assert high: when button E is released

	Assert low: when button E is pressed
F	Assert high: when button F is released Assert low: when button F is pressed
KEY	Assert high: when button of the analog joystick is released Assert low: when button of the analog joystick pressed

Table 3-6-1: Joystick shield module function description

After identify the pin function of the joystick shield module, we may proceed to the connection of the dsPIC30F4013 microcontroller with the joystick shield module. Warn that the joystick shield module has 2 analog output pin which are X and Y and both output will need the ADC of the microcontroller to handle. Following diagram shows the schematic of the connection of joystick shield module with dsPIC30F4013 microcontroller.

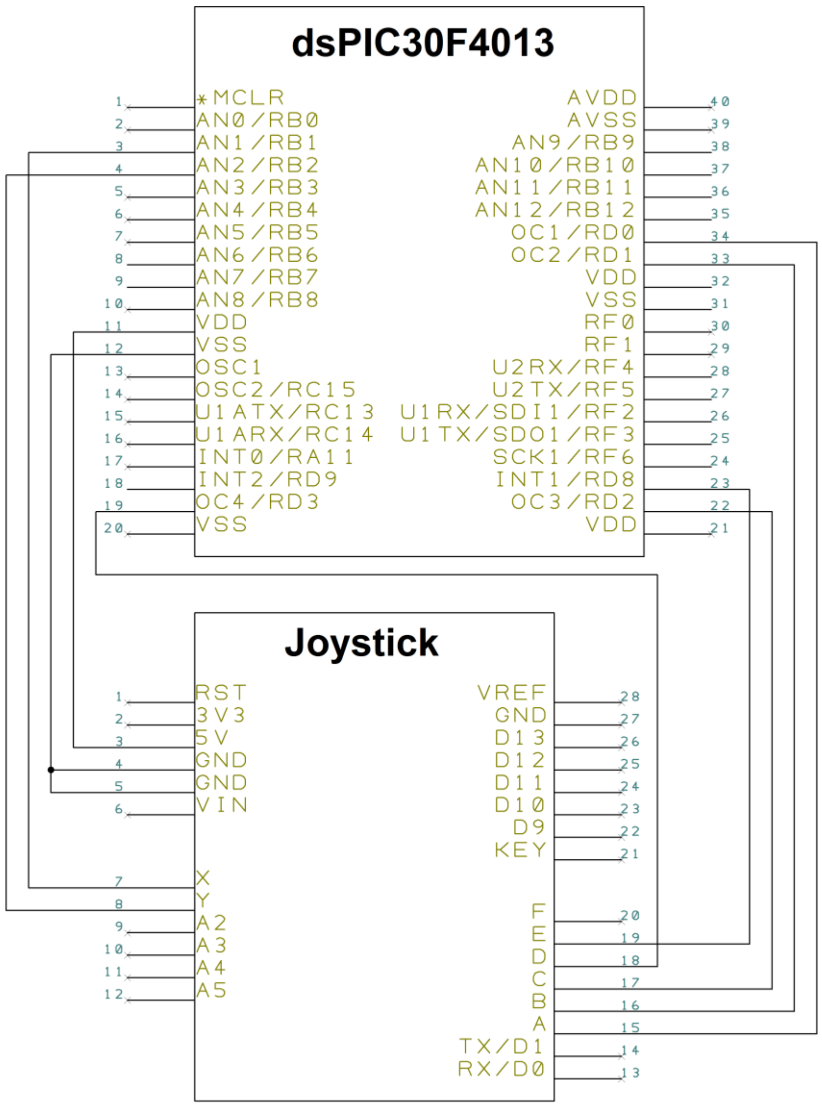


Figure 3-6-3: Schematic view of joystick controller

After complete the connection according to the schematic as above diagram, next thing is the software handling which will be discuss in section 4-7.

3.7 Wireless module interfacing

Before this section, the robotic hand, arm and car can be only control by the hard coded program code that has been burn into the microcontroller's flash memory. It can be describe as doing the task to a set of rules or it is fully control by the microcontroller's program, which is indirect control by human since the program code is develop by human. For direct control by human, specific interface need to be connected, for example: wired joystick, Bluetooth joystick, RF joystick, smartphone with Bluetooth connection or Wi-Fi connection capability, computer with Bluetooth connection or Wi-Fi connection capability and etc. In this project, a wireless joystick with sensor glove is used as controlling unit. Since it is wireless, there is several wireless communication protocols allow to do so, for example: Bluetooth, RFID, ZigBee, Wi-Fi and etc. RFID is choosing as the wireless communication protocol for this project and the chip module used is nRF24L01. Following diagram shows the chip diagram of the nRF24L01.

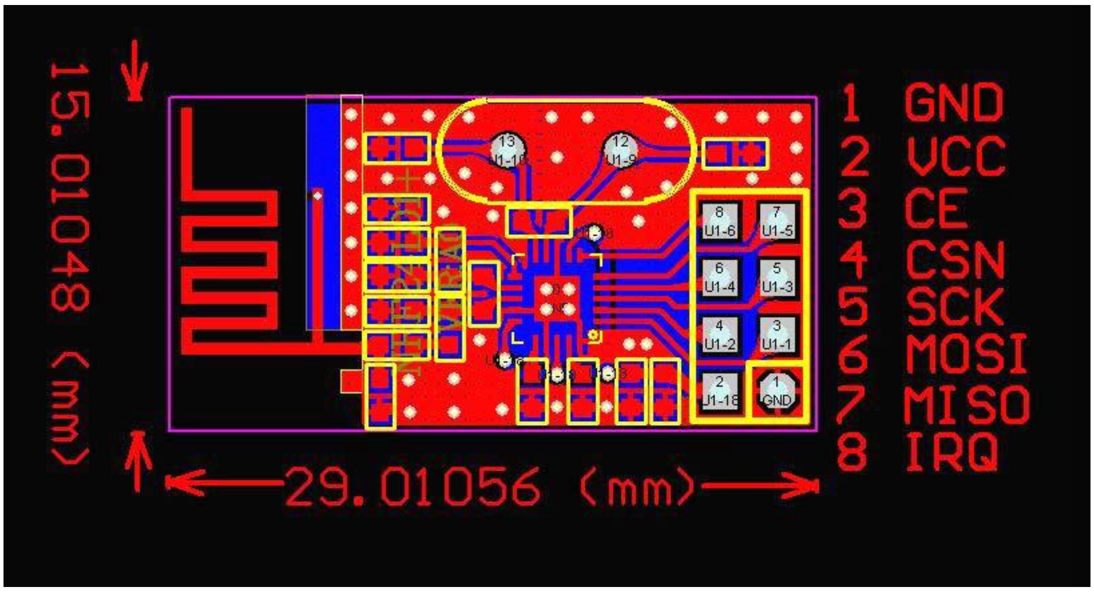


Figure 3-7-1: Pin diagram of nRF24L01

nRF24L01 transceiver module, which is a RF module that allows the data transfer to be in full duplex meaning that it can transmit and receive data simultaneously. Following table shows the specification of the nRf24L01 module.

Name	nRF24L01 transceiver
Mode	Transmitter + receiver
Voltage range	1.9 ~ 3.6 V
Frequency	2.4 GHz
Transfer rate	Up to 2 Mbps
Duplex mode	Full duplex
Communication interface	SPI

Table 3-7-1: Specification of the nRf24L01 module

With refers to the table above, the communication interface that used to connect the nRF24L01 with the microcontroller is the SPI peripheral. There is 1 SPI module integrated inside the dsPIC30F4013 and following diagram shows the connection of the nRF24L01 with the dsPIC30F4013 microcontroller.

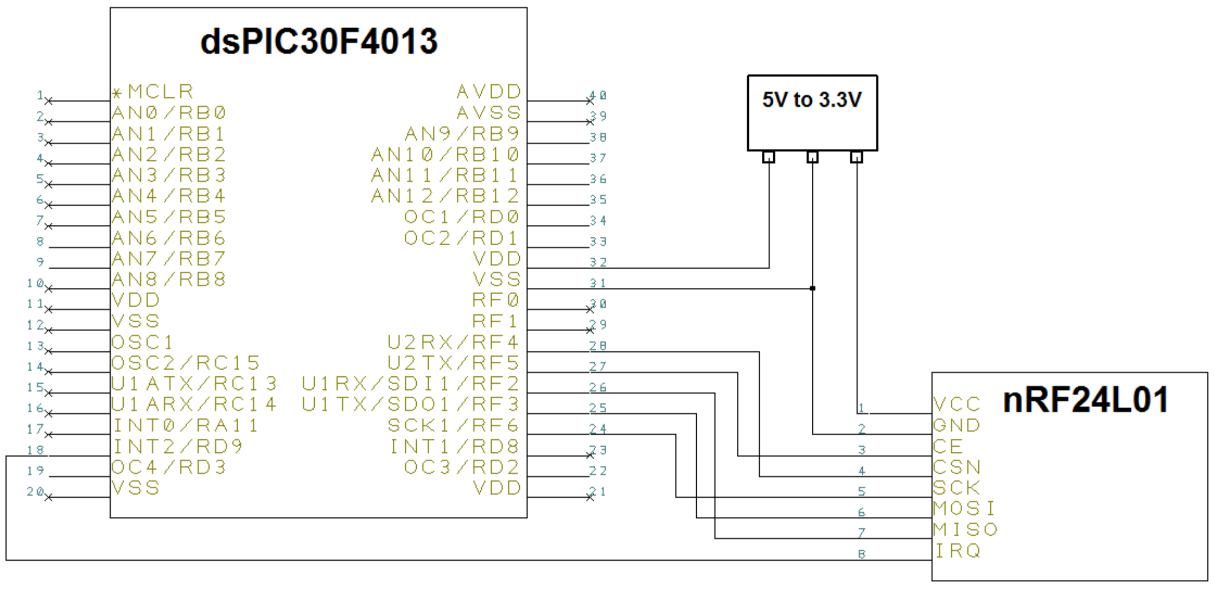


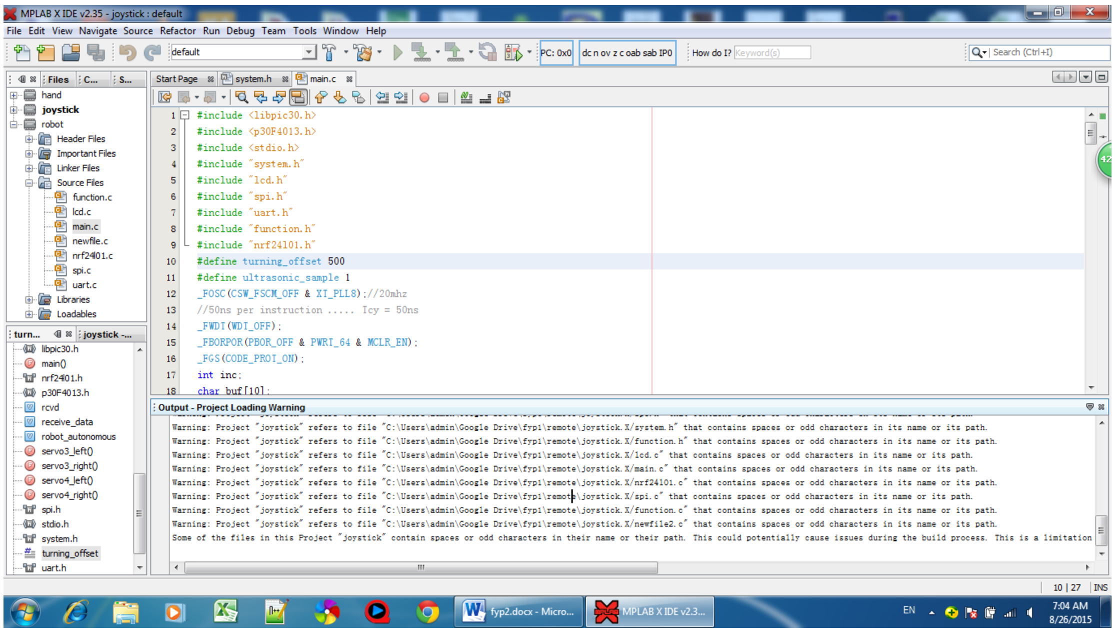
Figure 3-7-2: Schematic view of nRF24L01 with dsPIC30F4013

The next step to do is that the software handlings of the data transmit and receive and it will be discussed in section 5-2-2 - Wireless Packet Transmission Protocol.

Chapter 4 Software Development

4.1 Programming tool and equipment

In this project, the microcontroller used is PIC microcontroller and it is develop by Microchip Inc. To ensure developer can fully use their product, an IDE has introduced which is the MPLAB X IDE. This IDE allows developer to write the program code in C language and compile the code into the machine code. This IDE support a variety of microcontroller developed by Microchip Inc. However, to allow the C program code compile into machine code that follow the instruction sets of the microcontroller used in this project which is the dsPIC30F4013 , extra tool chains are needed to include which are the C30 compiler, ASM30 assembler and Link30 linker. Following diagram shows the MPLAB X IDE.

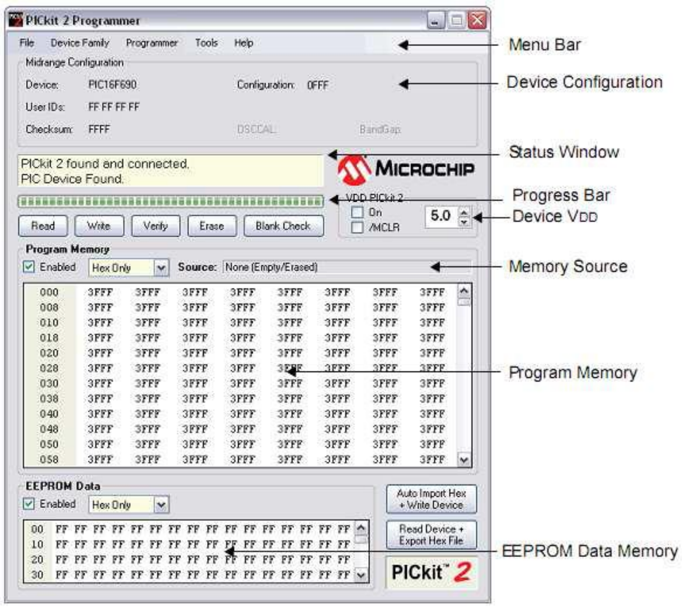


```
#include <libpic30.h>
#include <p30F4013.h>
#include <stdio.h>
#include "system.h"
#include "lcd.h"
#include "spi.h"
#include "uart.h"
#include "function.h"
#include "nrf24101.h"
#define turning_offset 500
#define ultrasonic_sample 1
_FOSC(CSW_FSCM_OFF & XT_PLL8);//20mhz
//50ns per instruction ..... Tcy = 50ns
_FWDT(WDT_OFF);
_FBORPOR(PBOR_OFF & PWRT_64 & MCLR_EN);
_FGS(CODE_PROT_ON);
int inc;
char buf[10];
```

```
Warning: Project "joystick" refers to file "C:\Users\admin\Google Drive\fyp1\remote\joystick.X/system.h" that contains spaces or odd characters in its name or its path.
Warning: Project "joystick" refers to file "C:\Users\admin\Google Drive\fyp1\remote\joystick.X/function.h" that contains spaces or odd characters in its name or its path.
Warning: Project "joystick" refers to file "C:\Users\admin\Google Drive\fyp1\remote\joystick.X/lcd.c" that contains spaces or odd characters in its name or its path.
Warning: Project "joystick" refers to file "C:\Users\admin\Google Drive\fyp1\remote\joystick.X/main.c" that contains spaces or odd characters in its name or its path.
Warning: Project "joystick" refers to file "C:\Users\admin\Google Drive\fyp1\remote\joystick.X/nrf24101.c" that contains spaces or odd characters in its name or its path.
Warning: Project "joystick" refers to file "C:\Users\admin\Google Drive\fyp1\remote\joystick.X/spi.c" that contains spaces or odd characters in its name or its path.
Warning: Project "joystick" refers to file "C:\Users\admin\Google Drive\fyp1\remote\joystick.X/function.c" that contains spaces or odd characters in its name or its path.
Warning: Project "joystick" refers to file "C:\Users\admin\Google Drive\fyp1\remote\joystick.X/newfile2.c" that contains spaces or odd characters in its name or its path.
Some of the files in this Project "joystick" contain spaces or odd characters in their name or their path. This could potentially cause issues during the build process. This is a limitation
```

Figure 4-1-1: MPLAB X IDE

After compile the code, the hex file will be store in [user_file]\[project_name].X\dist\ default\production\ [project_name].X.production. To burn the hex code to the microcontroller, a software tool and a hardware tool are needed which are PICkit2 and the UIC00B USB ICSP PIC programmer respectively. Following diagrams show the PICkit2 and the UIC00B programmer.



Midrange Configuration
Device: PIC16F690   Configuration: 0FFF
User IDs: FF FF FF FF
Checksum: FFFF   OSCCAL: BandGap:

PICkit 2 found and connected.
PIC Device Found.

VDD PICkit 2
☐ On
☐ /MCLR
5.0

Program Memory
☑ Enabled Hex Only Source: None (Empty/Erased)

```
000   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
008   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
010   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
018   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
020   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
028   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
030   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
038   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
040   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
048   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
050   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
058   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
```

EEPROM Data
☑ Enabled Hex Only

```
00  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
20  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
30  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Figure 4-1-2: PICkit2 programmer application

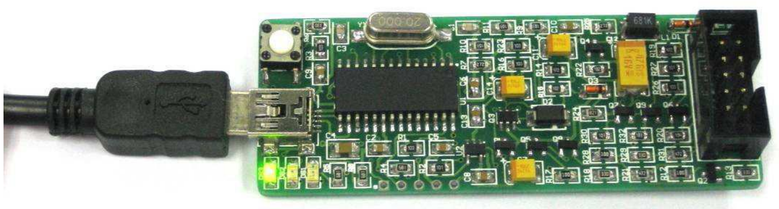
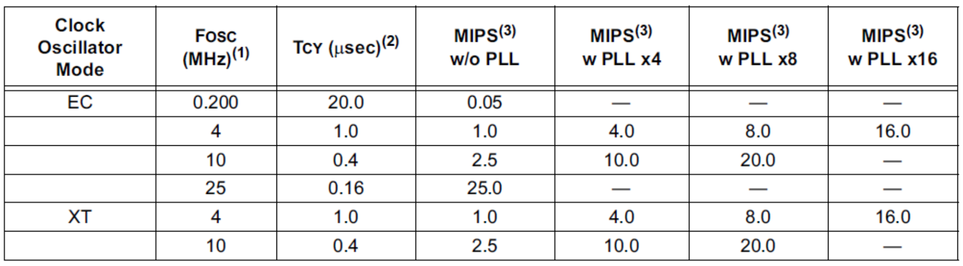


Figure 4-1-3: UIC00B programmer

4.2 System and timer frequency configuration

For this section, it is important that to pre-configure the system setting in order to turn the microcontroller going to live. First thing to do is by reading the datasheet written by the manufacturer. After read the datasheet, we need to define the system frequency. Based on the datasheet, the system frequency can be calculated based on following table.



Clock Oscillator Mode	Fosc (MHz)(1)	TCY (μsec)(2)	MIPS(3) w/o PLL	MIPS(3) w PLL x4	MIPS(3) w PLL x8	MIPS(3) w PLL x16
EC	0.200	20.0	0.05	—	—	—
	4	1.0	1.0	4.0	8.0	16.0
	10	0.4	2.5	10.0	20.0	—
	25	0.16	25.0	—	—	—
XT	4	1.0	1.0	4.0	8.0	16.0
	10	0.4	2.5	10.0	20.0	—

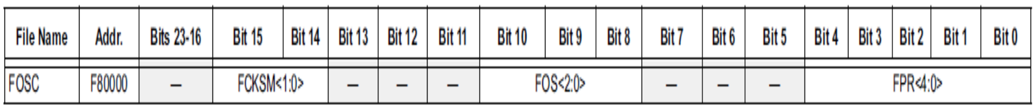
Remark: 1) EC represent external clock, XT represent external oscillator

2) MIPS represent million instructions per second

Table 4-2-1: dsPIC30F4013 clock frequency table

The system performance is define by how much MIPS that system able to perform. To calculate the MIPS of dsPIC30F4013, first, we need to identify the clock source speed, for example, we used an external oscillator with 10 MHz. The 10Mhz is then divide by 4 and multiply with the prescaler value, for example, 8 times prescaler produce (10MHz / 4) x 8 = 20Mhz, which means that the microcontroller is able 20 MIPS with each clock cycle time of 50 nanoseconds.

Since the clock source for the microcontroller used in this project is generated by external oscillator and we desire to maximum use for the performance design with the microcontroller which is 20 MIPS. In order to achieve it, the external oscillator we used is 10MHz and the prescaler value is set to 8 times. Next, to allow such condition occurs, register FOSC need to be rewrite for every time microcontroller power on in which the C program code can handle for this situation. Following diagram shows the FOSC register and the table shows the register function description.



File Name	Addr.	Bits 23-16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FOSC	F80000	—	FCKSM<1:0>		—	—	—	FOS<2:0>			—	—	—	FPR<4:0>				

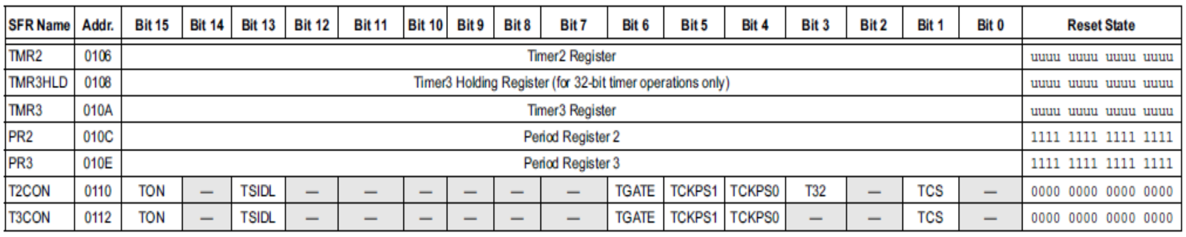
Figure 4-2-1: FOSC register

FCKSM	00	Switching between different sources of the clock is enabled. Clock monitoring is enabled.
	01	Switching between different sources of the clock is enabled. Clock monitoring is disabled.
	1x	Switching between different sources of the clock is disabled. Clock monitoring is disabled.
FOS	00	Low power 32kHz internal oscillator (timer 1)
	01	Internal fast RC oscillator
	10	Internal low power RC oscillator
	11	Primary oscillator selected by FPR
FPR	000x	XTL
	001x	HS
	0100	XT
	0101	XT PLL 4x
	0110	XT PLL 8x
	0111	XT PLL 16x
	1000	ERCIO - OSC2 pin is I/O
	1001	ERC - OSC2 pin is system clock output (FOSC/4)
	1010	Reserved
	1011	EC
	1100	ECIO
	1101	EC PLL 4x
	1110	EC PLL 8x
	1111	EC PLL 16x

Table 4-2-2: FOSC register function table

For our case, in which as previous stated to achieve the highest performance of the dsPIC30F4013 microcontroller that is 20 MIPS where the clock source is generate by the external oscillator which 10 MHz frequency, the FOS is set to 11 and the FPR is set to 0110.

While for the timer frequency setting, let’s assume the timer to set is TIMER2 since all of the timers inside the microcontroller are same in configuration. For TIMER2, it is consists of 16-bit value of the period, which means that it is count from 0 to 65535 and it is able to count for 65536 cycles and then reset back to 0 and start another loop of count. From previous system frequency, 1 clock cycle is equivalent to 50 nanoseconds, which means for a full period of TIMER2 count is equivalent to 3.2768 milliseconds. However, TIMER2 will be used to generate the PWM signal for the servo motor. The working frequency for the servo motor is limit to 50Hz, which equivalent to 20 milliseconds. In order to fit the TIMER2 with the requirement of the servo motor, the TIMER2 need to be prescaled another time to make a slower clock cycle, warn that slowing down the TIMER2 clock cycle will not affect to the system clock cycle time. Diagram below shows the register set for configuring the TIMER2 and its functional description.



SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
TMR2	0106	Timer2 Register																uuuu uuuu uuuu uuuu
TMR3HLD	0108	Timer3 Holding Register (for 32-bit timer operations only)																uuuu uuuu uuuu uuuu
TMR3	010A	Timer3 Register																uuuu uuuu uuuu uuuu
PR2	010C	Period Register 2																1111 1111 1111 1111
PR3	010E	Period Register 3																1111 1111 1111 1111
T2CON	0110	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	T32	—	TCS	—	0000 0000 0000 0000
T3CON	0112	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	—	TCS	—	0000 0000 0000 0000

Figure 4-2-2: TIMER2 module register set

Register	Function name	Selection	Function
TIM2			Store TIMER2 counting value with each count represent 1 cycle
PR2			Store the maximum value of the TIMER2, TIM2

			will be reset back to 0 after reach PR2
T2CON	TON	0	Stop the operation of TIMER2 module
		1	Start the operation of TIMER2 module
	TSIDL	0	Continue operation of TIMER2 when in IDLE mode
		1	Stop operation of TIMER2 when in IDLE mode
	TGATE	0	Timer gated time accumulation mode disable
		1	Timer gated time accumulation mode enable
	TCKPS	00	1:1 prescale value
		01	1:8 prescale value
		10	1:64 prescale value
		11	1:256 prescale value
	T32		Reserved for 32-bit count (join with TIMER3)
	TCS	0	Clock source from internal clock
		1	Clock source from external clock from pin T2CK

Table 4-2-3: TIMER2 module register set function table

To meet the requirement, the TIMER2 is reconfigure by set the TCKPS to 01, which is 8 system clock cycle represent TIMER2's TIM2 1 count. Originally, the system clock cycle time is 50 nanoseconds, whereby after prescaled for the TIMER2, 50 nanoseconds is multiply by 8 and 400 nanoseconds is represent 1 clock cycle time of the TIMER2. Thus, for a full period count of TIMER2, which is 65536 multiply with 400 nanoseconds, is equivalent to 26.2144 milliseconds.

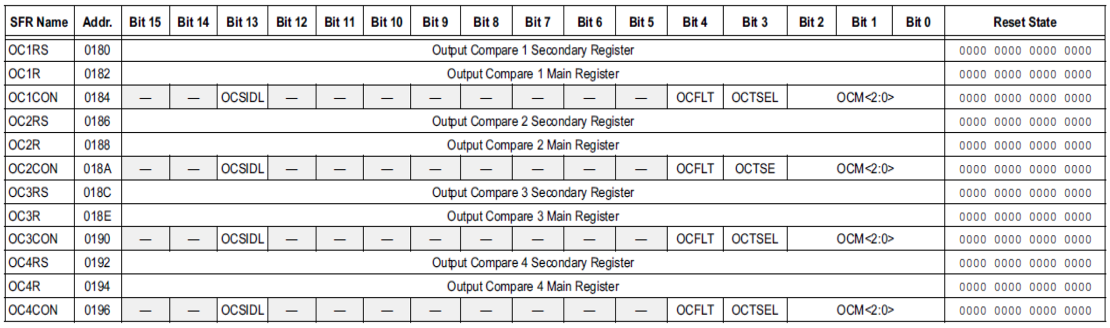
4.3 Robotic car programming

With refers to section 3-2, in this section, we will develop a program code that is able to control the robotic car to movement to desire direction. As stated in section 3.2, a total of 2 PWM signal and 4 GPIO signals need to be generate out from the dsPIC30F4013 microcontroller. The first thing to identify is that the speed of the robotic car, in which is define by the rotation speed of the dc motor. The PWM signals generated by the microcontroller can be used to control the rotation speed of the dc motor in terms of the duty cycle produce. By assuming the dc motor can perform with a period of 200 Hz, use TIMER2 to generate the PWM signal and using the timer setting as discuss in section 4-2, the formula to calculate the PR2 of the TIMER2 is shown below:

$$PWM\ period = (PR2 + 1)\ \times\ System\ clock\ \times\ TMR2\ prescaler$$

where 1)	PWM period	= 1/200Hz
		= 5 milliseconds
2)	FOSC	= 20 MHz
3)	System clock	= 50 nanoseconds
4)	TMR2 prescaler	= 8

According to the formula above, the calculated PR2 is 12499, which is 30D3H in hex value. Next is to identify the duty cycle in which duty cycle will affect the speed of the dc motor. The duty cycle is defined as the percentage of high logic over the each period. For example, in the case of 200 Hz period which is 5 milliseconds per period, if we choose to have 50% of duty cycle, then the logic high will keep for 2.5 milliseconds each period and another 2.5 milliseconds for logic low. In order to maximize the speed of the dc motor, we define 99% as the duty cycle of the PWM signal generated and 99% of duty cycle represent as 12375, which is 3057H in hex value. After a series of calculation, now is the time to develop the program code so that the robotic car can perform as stated previous by set or clear the register value of the register file inside the dsPIC30F4013 microcontroller. Following diagram shows the related register file that require for this section and the register function description.



SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
OC1RS	0180	Output Compare 1 Secondary Register																0000 0000 0000 0000
OC1R	0182	Output Compare 1 Main Register																0000 0000 0000 0000
OC1CON	0184	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000 0000 0000 0000
OC2RS	0186	Output Compare 2 Secondary Register																0000 0000 0000 0000
OC2R	0188	Output Compare 2 Main Register																0000 0000 0000 0000
OC2CON	018A	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSE	OCM<2:0>			0000 0000 0000 0000
OC3RS	018C	Output Compare 3 Secondary Register																0000 0000 0000 0000
OC3R	018E	Output Compare 3 Main Register																0000 0000 0000 0000
OC3CON	0190	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000 0000 0000 0000
OC4RS	0192	Output Compare 4 Secondary Register																0000 0000 0000 0000
OC4R	0194	Output Compare 4 Main Register																0000 0000 0000 0000
OC4CON	0196	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000 0000 0000 0000

Figure 4-3-1: Output compare module register set

Register	Function name	Selection	Function
OCxRS			Store the duty cycle value – Secondary
OCxR			Store the duty cycle value – Main
OCxCON	OCSIDL	0	active in IDLE state
		1	inactive in IDLE state
	OCFLT	0	no FAULT occurred
		1	FAULT occurred, hardware reset only
	OCTSEL	0	TIMER2 selected
		1	TIMER3 selected
	OCM	000	Output Compare Module disabled
		001	Single compare match mode, pin OCx driven high
		010	Single compare match mode, pin OCx driven low
		011	Single compare match mode, pin OCx toggles
		100	Dual compare match mode, single output pulse at pin OCx
		101	Dual compare match mode, sequence of output pulses at pin OCx
		110	PWM mode without fault protection input

		111	PWM mode with fault protection input

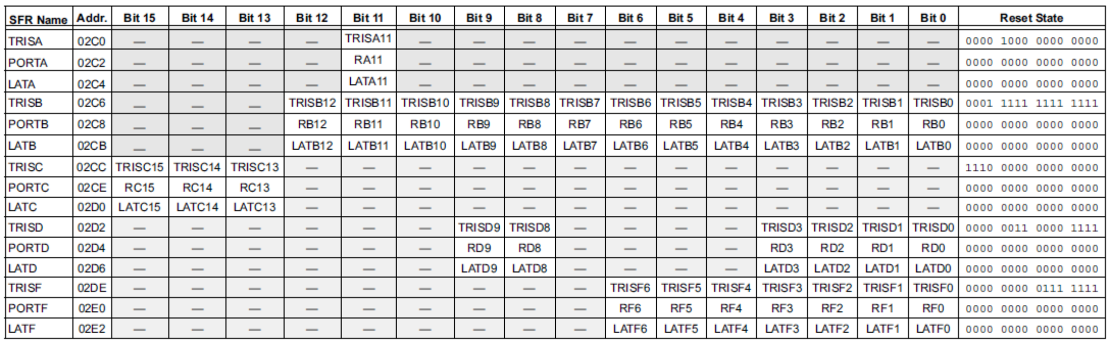
x = 1, 2, 3 and 4

Table 4-3-1: Output compare module register set function table

With refers to the requirement and the register set of the microcontroller, the steps of setting the register value are as below:

1) Set PR2 = 0x30D3
2) Set OC3CON and OC4CON to 0x0006 since the robotic car PWM signal pin is connected to the OC3 and OC4. 0x0006 is represent 0000 0000 0000 0110 in binary form and is setting the OC3 and OC4 to active in IDLE state, no FAULT occurred, TIMER2 as clock source and select PWM mode without fault protection input.
3) Set TCKPS of T2CON discussed in previous section to 01 to indicate 1:8 prescale values.
4) Start the TIMER2

In order to control the direction of the robotic car, the GPIO pin that connected to the L298N must be set or clear according to direction wanted. Before assigned any value to the GPIO register, it must be indicate the direction of the GPIO, either input or output through the TRISx register, where x = A, B, C, D and F. The related register file that require by the GPIO is shown in the diagram below.



SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
TRISA	02C0	—	—	—	—	TRISA11	—	—	—	—	—	—	—	—	—	—	—	0000 1000 0000 0000
PORTA	02C2	—	—	—	—	RA11	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000
LATA	02C4	—	—	—	—	LATA11	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000
TRISB	02C6	—	—	—	TRISB12	TRISB11	TRISB10	TRISB9	TRISB8	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	0001 1111 1111 1111
PORTB	02C8	—	—	—	RB12	RB11	RB10	RB9	RB8	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	0000 0000 0000 0000
LATB	02CB	—	—	—	LATB12	LATB11	LATB10	LATB9	LATB8	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0	0000 0000 0000 0000
TRISC	02CC	TRISC15	TRISC14	TRISC13	—	—	—	—	—	—	—	—	—	—	—	—	—	1110 0000 0000 0000
PORTC	02CE	RC15	RC14	RC13	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000
LATC	02D0	LATC15	LATC14	LATC13	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000
TRISD	02D2	—	—	—	—	—	—	TRISD9	TRISD8	—	—	—	—	TRISD3	TRISD2	TRISD1	TRISD0	0000 0011 0000 1111
PORTD	02D4	—	—	—	—	—	—	RD9	RD8	—	—	—	—	RD3	RD2	RD1	RD0	0000 0000 0000 0000
LATD	02D6	—	—	—	—	—	—	LATD9	LATD8	—	—	—	—	LATD3	LATD2	LATD1	LATD0	0000 0000 0000 0000
TRISF	02DE	—	—	—	—	—	—	—	—	—	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	TRISF0	0000 0000 0111 1111
PORTF	02E0	—	—	—	—	—	—	—	—	—	RF6	RF5	RF4	RF3	RF2	RF1	RF0	0000 0000 0000 0000
LATF	02E2	—	—	—	—	—	—	—	—	—	LATF6	LATF5	LATF4	LATF3	LATF2	LATF1	LATF0	0000 0000 0000 0000

Figure 4-3-2: GPIO module register set

To be able to make a pin as output TRISx of the related pin must be clear or set to indicate as input. The initial values of the TRISx registers are all set, it is indicate that all the pins act as input after a reset. To be able to use the related pins to control the direction of the robotic car, which are RB9, RB10, RB11 and RB12 with latches output, the register file corresponding are TRISB and LATB. The TRISB register is written with 0x0000 and following table shows the value of LATB of each direction.

Direction	Pin				LATB value
	RB9	RB10	RB11	RB12	
Forward	0	1	0	1	0x1400
Reverse	1	0	1	0	0x0A00
Left	0	1	1	0	0x0C00
Right	1	0	0	1	0x1200
Left- Forward	0	0	0	1	0x1000
Right - Forward	0	1	0	0	0x0400
Left- Reverse	0	0	1	0	0x0800
Right - Reverse	1	0	0	0	0x0200

Table 4-3-2: Direction control code table

Another feature can be implementing is that when the TMR2 of TIMER2 equals to PR2, overflow occurs and TIMER2 interrupt flag will be set by the microcontroller. Although the overflow of TIMER2 will be not affect the signal generated in which TMR2 will be reset back to 0x0000 after reach PR2. But for good programming practices and to ensure no any runtime error happen while microcontroller in execution, extra codes need to be added. The extra code is that the Interrupt Service Routine (ISR) portion of code. These portions of code perform no other than clearing the overflow flag of the related timer and start back the timer again. The flow chart below shows the program flow that had been developed to handle and generate the PWM signal with extra ISR handling for the robotic car.

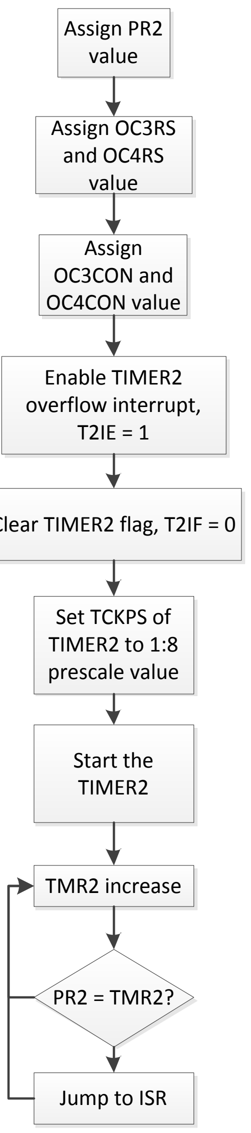

Assign PR2 value
Assign OC3RS and OC4RS value
Assign OC3CON and OC4CON value
Enable TIMER2 overflow interrupt, T2IE = 1
Clear TIMER2 flag, T2IF = 0
Set TCKPS of TIMER2 to 1:8 prescale value
Start the TIMER2
TMR2 increase
PR2 = TMR2?
Jump to ISR

Figure 4-3-3: Robotic car PWM signal generation program flow

The direction of moving of the robotic car can be directly control by amend the LATB register as stated in the previous table.

4.4 Robotic hand programming

With proceed with the TIMER configuration setting as discussed in the section 4-2, now is to develop the program flow and code for the robotic hand. Before develop the code, some basic theory need to learn first that is the working principle of the servo motor since there are 5 servo motor used to control the robotic fingers bending degree. The following diagram shows the basic working theory of a servo motor in which the timing requirement of the PWM signal generated.

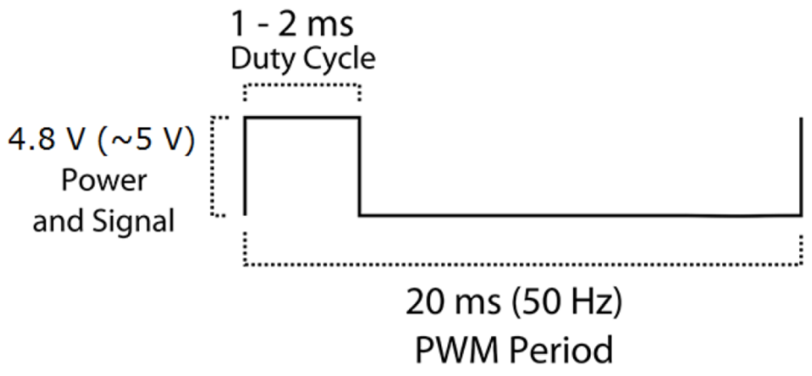


Figure 4-4-1: Servo motor timing requirement

For a servo motor to work, the basic requirement is that generated a signal of 20 milliseconds of period with consists of 1 to 2 milliseconds of duty cycle. However, note that not every servo motor will had exactly the same duty cycle range, even for the same brand and model. Due to this reason, every servo motor used in this project needs to have testing on the duty cycle range in order to accurately control on the rotating degree. To test for the duty cycle range, the first to do is to develop the test code and the test code will also be reused for the developing the actual program code. By the end of last section which is section 4-3 – robotic car programming, we would be able to generate the PWM signal. However, in order to generate the accurate duty cycle, a series of calculation needs to be taken. Following shows the working steps to define the duty cycle of the servo motor to perform, assume that the duty cycle range is from 1 to 2 milliseconds.

1) Using the formula below to calculate the duty cycle and the value calculated will be write into OCxRS register, where x=1, 2, 3, 4

$$Duty\ cycle = (OCxRS + 1) \times System\ clock \times TMR2\ prescaler$$

2) Let duty cycle = 1 millisecond

$$1 \times 10^{-3} = (OCxRS + 1) \times (50 \times 10^{-9}) \times 8$$

$$OCxRS = 2499$$

3) Let duty cycle = 2 millisecond

$$2 \times 10^{-3} = (OCxRS + 1) \times (50 \times 10^{-9}) \times 8$$

$$OCxRS = 4999$$

4) The OCxRS = 2499 represent for 1 millisecond duty cycle while OCxRS = 4999 represent for 2 millisecond duty cycle

5) The period of the PWM signal are 20 milliseconds and using following formula to calculate the value that need to write into the PR2 register since the PWM signals using TIMER2 as based clock.

$$PWM\ period = (PR2 + 1) \times System\ clock \times TMR2\ prescaler$$

6) From the formula given, the period value equals to 49999 and will be write into PR2 register.

Using the value calculated as above, the program code will be developing based on the following flow chart.

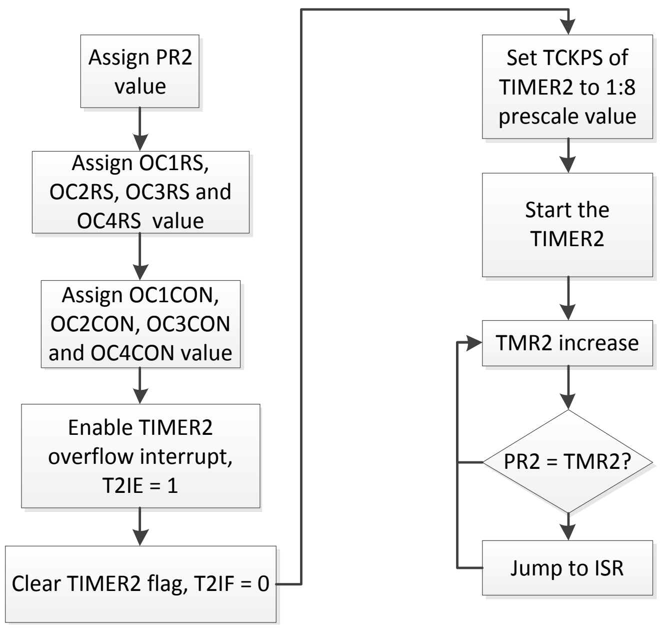
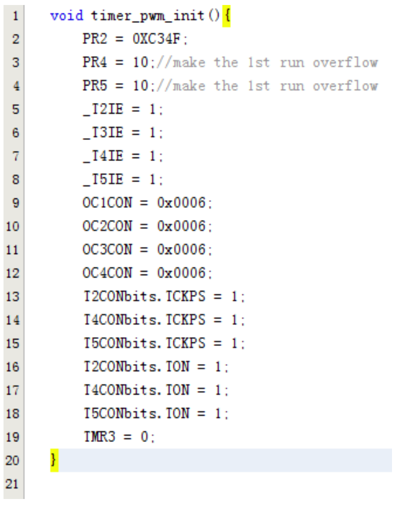

Assign PR2 value
Assign OC1RS, OC2RS, OC3RS and OC4RS value
Assign OC1CON, OC2CON, OC3CON and OC4CON value
Enable TIMER2 overflow interrupt, T2IE = 1
Clear TIMER2 flag, T2IF = 0
Set TCKPS of TIMER2 to 1:8 prescale value
Start the TIMER2
TMR2 increase
PR2 = TMR2?
Jump to ISR

Figure 4-4-2: Robotic hand PWM signal generation program flow

Since the robotic hand has 5 fingers and the PWM output of the dsPIC30F4013 microcontroller can only support for 4 PWM output. Therefore, a special method with a portion of extra code is developing to handle for such condition. Originally, there are 2 type of method to generate an extra PWM signal, one is by polling method, and another is by interrupt method.

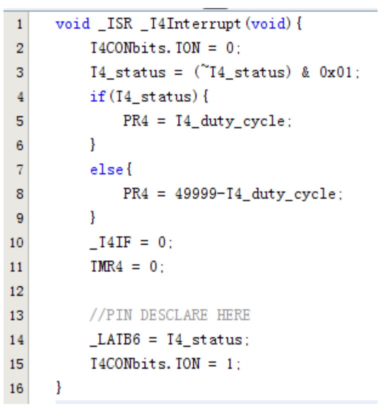
Polling method is that using a system timer delay to generate the PWM signal and it must be always inside the main loop. For this situation, if a main loop has 10 milliseconds of execution time without generating a PWM signal, then after using the polling method, the execution time of the main loop will be increase to 30 milliseconds since the period of the servo motor is 20 milliseconds. This method is not very efficient in terms of it will create another delay for the main loop. For another view, if a main loop has 100 milliseconds of execution time without generating a PWM signal, then after using the polling method to generate the PWM signal, the execution time of the main loop will be in a total of 120 milliseconds. In such situation, the servo motor period will be increase to 120 milliseconds too due to it is depends on the sequential execution of the main loop. This may be disastrous when the robotic hand is holding an object; the servo motor will only work for 20 milliseconds then idle for 100 milliseconds and that will make the robotic finger to suddenly relax and the object may fall.

To avoid such condition happen, therefore the second method is used which is the interrupt method. The following diagrams show the program code of using interrupt method to produce an extra PWM signal.



```
void timer_pwm_init(){
    PR2 = 0XC34F;
    PR4 = 10;//make the 1st run overflow
    PR5 = 10;//make the 1st run overflow
    _T2IE = 1;
    _T3IE = 1;
    _T4IE = 1;
    _T5IE = 1;
    OC1CON = 0x0006;
    OC2CON = 0x0006;
    OC3CON = 0x0006;
    OC4CON = 0x0006;
    T2CONbits.TCKPS = 1;
    T4CONbits.TCKPS = 1;
    T5CONbits.TCKPS = 1;
    T2CONbits.TON = 1;
    T4CONbits.TON = 1;
    T5CONbits.TON = 1;
    TMR3 = 0;
}
```

Figure 4-4-3: Extra PWM signal generation initialization code



```
void _ISR _T4Interrupt(void){
    T4CONbits.TON = 0;
    T4_status = (~T4_status) & 0x01;
    if(T4_status){
        PR4 = T4_duty_cycle;
    }
    else{
        PR4 = 49999-T4_duty_cycle;
    }
    _T4IF = 0;
    TMR4 = 0;

    //PIN DESCLARE HERE
    _LATB6 = T4_status;
    T4CONbits.TON = 1;
}
```

Figure 4-4-4: Extra PWM signal generation ISR handling code

Notice that we use TIMER4 to generate an extra PWM signal and output the signal through RB6 pin of the dsPIC30F4013 microcontroller. From the 1st diagram, line 4 showing that PR4 = 10, it is purposely to make it overflow and jump to ISR when timer start. Note that must enable the interrupt of the TIMER4 then only start the timer. Then from 2nd diagram, which is the ISR handling code of the TIMER4. Line 3 of the code is used to invert the status of the TIMER4, logic 0 for to perform the logic low of the period while logic 1 to perform the duty cycle part. With the status inverting, it will also refer as the output at RB6 pin with latch output, LATB6. Line 4 to 9 of the codes had shown the condition to handle when the status changed. Focus on line 8, 49999 is the value represent the period of the servo motor. The timer used is TIMER4 with prescale value of 8 and thus each clock cycle time represent for 400 nanoseconds. The period and the duty cycle calculation will be same as previous stated. The duty cycle of the PWM signal generated to RB6 pin will be define by T4_duty_cycle variable, which is a global variable that declare in the main code file.

4.5 Robotic arm programming

For the robotic arm programming, first thing to do is to identify the part and the axes that control by servo and named each servo to avoid confusing when develop the program code. Following diagram shows axis and the part control by each servo and the table shows the degree of rotation, the microcontroller connected with and the extra information of each servo.

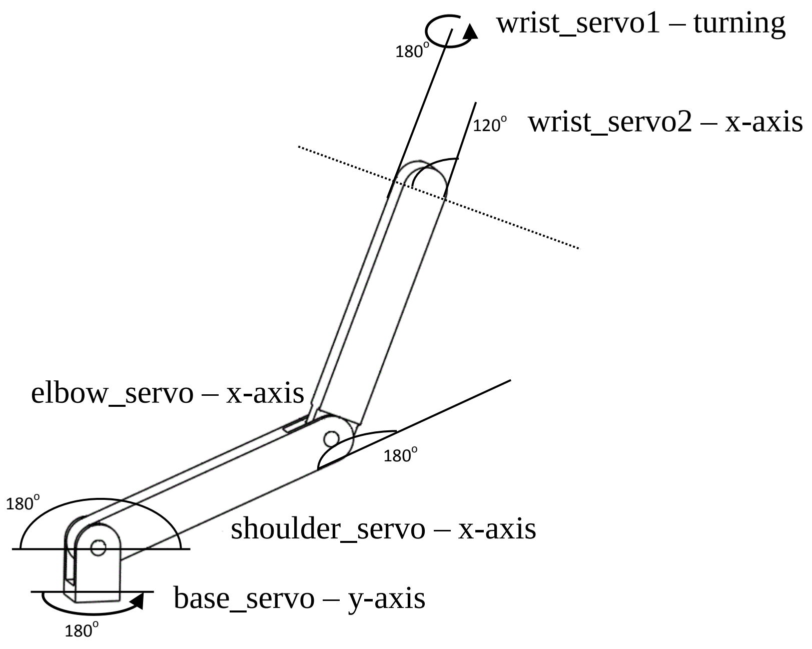


Figure 4-5-1: Servo motor control axis and naming convention

Servo name	Rotation Degree	Microcontroller	Pin	Remark (PWM signal generated by)
wrist_servo1	0 – 180°	Robotic hand's	RB8	TIMER5
wrist_servo2	0 – 120°	Robotic car's	RB7	TIMER5
elbow_servo	0 – 180°	Robotic car's	RB6	TIMER4
shoulder_servo	0 – 180°	Robotic car's	RD1	PWM module

base_servo	0 – 180^o	Robotic car’s	RD0	PWM module

Table 4-5-1: Servo motor part control and information

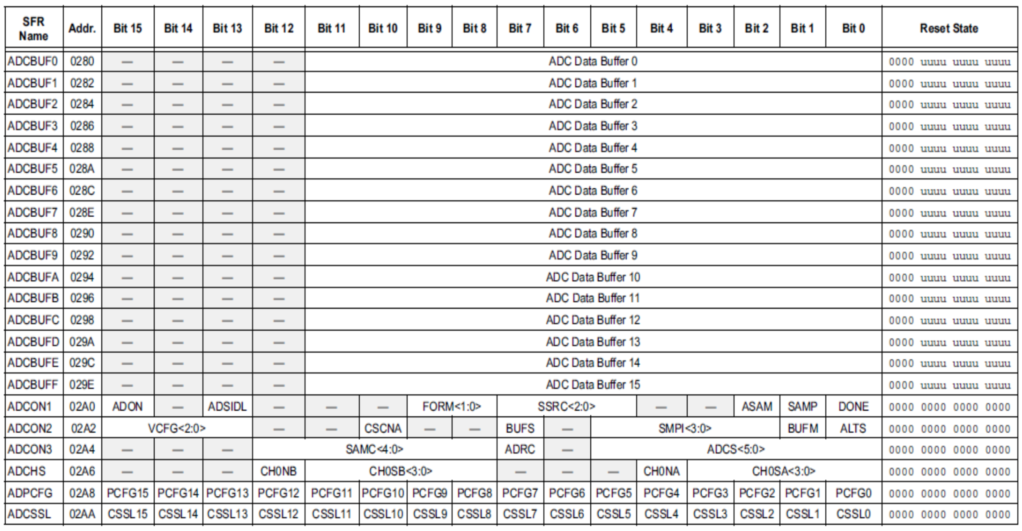
Refers to the table, the PWM signal generated by TIMER4 and TIMER5 had been discussed in the previous section, which using timer to generate signal and using interrupt method to change signal for duty-cycle part and non- duty-cycle part.

4.6 Sensor glove

With the complete circuit design as shown in section 3.5, now we will focus on the software program of the sensor glove. To start for developed the program code, the ADC module of the dsPIC30F4013 microcontroller in order to write the simple program for ADC to get the data and process the data. The ADC module integrated in the dsPIC30F4013 microcontroller is a 12-bit analog-to-digital converter which means that the sample value will be in the digital form value maximum to 4095. By refer to the circuit of the sensor glove shown in section 3.5, the sensor glove is supply with 5V of voltage supply, which means the voltage range the ADC can sample from sensor glove is limit with the range of 0 to 5V. Thus, with 5V divide by 4096 steps, each sampling step will contribute of 1.22×10^{-3} V, which is very high, precise and accurate in analog voltage sampling. Next, in order to develop the program code with proper sample the data and use the data to perform something, there are no other than reading the datasheet of the dsPIC30F4013 microcontroller. Below steps shows the simplify way to sample the data using ADC of the dsPIC30F4013 microcontroller.

1) Configure the ADC module:
   - Configure analog pins, voltage reference and digital I/O
   - Select ADC input channels, conversion clock, conversion trigger
   - Turn on ADC module
2) Start sampling
3) Wait the required acquisition time which simply do a system delay of 5 microseconds
4) Trigger acquisition end, start conversion:
5) Wait for ADC conversion to complete, by waiting for the DONE bit to get set.
6) Read ADC result buffer

With refers to the above steps, now we will investigate on the development of the program code. Diagram below shows the register set that need by ADC module and the functional description of each register.



SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
ADCBUF0	0280	—	—	—	—	ADC Data Buffer 0												0000 uuuu uuuu uuuu
ADCBUF1	0282	—	—	—	—	ADC Data Buffer 1												0000 uuuu uuuu uuuu
ADCBUF2	0284	—	—	—	—	ADC Data Buffer 2												0000 uuuu uuuu uuuu
ADCBUF3	0286	—	—	—	—	ADC Data Buffer 3												0000 uuuu uuuu uuuu
ADCBUF4	0288	—	—	—	—	ADC Data Buffer 4												0000 uuuu uuuu uuuu
ADCBUF5	028A	—	—	—	—	ADC Data Buffer 5												0000 uuuu uuuu uuuu
ADCBUF6	028C	—	—	—	—	ADC Data Buffer 6												0000 uuuu uuuu uuuu
ADCBUF7	028E	—	—	—	—	ADC Data Buffer 7												0000 uuuu uuuu uuuu
ADCBUF8	0290	—	—	—	—	ADC Data Buffer 8												0000 uuuu uuuu uuuu
ADCBUF9	0292	—	—	—	—	ADC Data Buffer 9												0000 uuuu uuuu uuuu
ADCBUFA	0294	—	—	—	—	ADC Data Buffer 10												0000 uuuu uuuu uuuu
ADCBUFB	0296	—	—	—	—	ADC Data Buffer 11												0000 uuuu uuuu uuuu
ADCBUFC	0298	—	—	—	—	ADC Data Buffer 12												0000 uuuu uuuu uuuu
ADCBUFD	029A	—	—	—	—	ADC Data Buffer 13												0000 uuuu uuuu uuuu
ADCBUFE	029C	—	—	—	—	ADC Data Buffer 14												0000 uuuu uuuu uuuu
ADCBUFF	029E	—	—	—	—	ADC Data Buffer 15												0000 uuuu uuuu uuuu
ADCON1	02A0	ADON	—	ADSIDL	—	—	—	FORM<1:0>		SSRC<2:0>			—	—	ASAM	SAMP	DONE	0000 0000 0000 0000
ADCON2	02A2	VCFG<2:0>			—	—	CSCNA	—	—	BUFS	—	SMPI<3:0>				BUFM	ALTS	0000 0000 0000 0000
ADCON3	02A4	—	—	—	SAMC<4:0>					ADRC	—	ADCS<5:0>						0000 0000 0000 0000
ADCHS	02A6	—	—	—	CH0NB	CH0SB<3:0>				—	—	—	CH0NA	CH0SA<3:0>				0000 0000 0000 0000
ADPCFG	02A8	PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8	PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0	0000 0000 0000 0000
ADCSSL	02AA	CSSL15	CSSL14	CSSL13	CSSL12	CSSL11	CSSL10	CSSL9	CSSL8	CSSL7	CSSL6	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0	0000 0000 0000 0000

Figure 4-6-1: ADC module register set

Register	Function name	Selection	Function
ADCBUFx			ADC sample buffer
ADCON1	ADON	0	ADC off
		1	ADC on
	ASIDL	0	continue operation in IDLE mode
		1	stop operation in IDLE mode
	FORM	00	Integer
		01	Signed integer
		10	Fractional
		11	Signed fractional
	SSRC	000	Clear SAMP bit to stop sampling and start conversion
		001	Transition on INT0 pin to stop sampling and start conversion
		010	TIMER3 compare to stop sampling and start conversion
		011	motor control PWM interval to stop sampling and

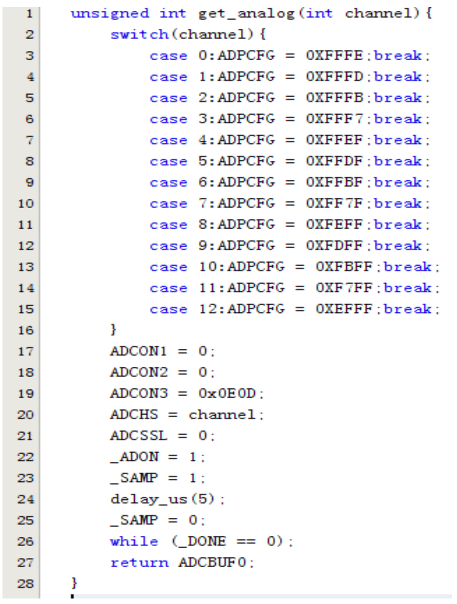
			start conversion
		10x	Reserved
		110	Reserved
		111	Automatic mode
	ASAM	0	Disable ADC sample auto-START bit
		1	Enable ADC sample auto-START bit
	SAMP	0	Stop sampling
		1	Start sampling
	DONE		Trigger by microcontroller, DONE bit will be set when A/D conversion is done
ADCON2	VCFG	000	VREFH =AVDD, VREFL=AVSS
		001	VREFH= External VREF+ pin, VREFL=AVSS
		010	VREFH =AVDD, VREFL=External VREF- pin
		011	VREFH= External VREF+ pin, VREFL=External VREF- pin
		1xx	VREFH =AVDD, VREFL=AVSS
	CSCNA		Set to enable the CH0 channel inputs to be scanned across a selected number of analog input
	BUFS		Trigger by microcontroller. Set when ADC currently filling higher buffer and clear when ADC currently filling lower buffer
	SMPI	xxxx	Generate interrupt after xxxx of bits
	BUFM	0	Buffer configured as one 16-word buffer ADCBUF
		1	Buffer configured as two 8-word buffer
	ALTS	0	Used MUX input
		1	alternate sampling enabled,
ADCON3	SAMC	xxxxx	Auto sample time bits
	ADRC	0	Clock source from system clock
		1	Clock source from internal RC clock
	ADCS	xxxxxx	ADC conversion clock

ADCHS	CH0NB	x	MUX B multiplexer setting bit
	CH0SB	xxxx	input select for MUX B multiplexer
	CH0NA	x	MUX A multiplexer setting bit
	CH0SA	xxxx	input select for MUX A multiplexer
ADPCFG	PCFGx		Set to indicate digital I/O, clear to indicate analog input pin
ADCSSL	CSSLx		pin scan selection bits

where x = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F

Table 4-6-1: ADC module register set function table

The formula to calculate the conversion clock is given by TAD$= TCY \times (0.5 \times (ADCS + 1))$, where TAD = ADC conversion time, TCY = system per clock cycle time. By using all the information as stated above, following program code had been developed.



```
unsigned int get_analog(int channel){
    switch(channel){
        case 0:ADPCFG = 0XFFFE;break;
        case 1:ADPCFG = 0XFFFD;break;
        case 2:ADPCFG = 0XFFFB;break;
        case 3:ADPCFG = 0XFFF7;break;
        case 4:ADPCFG = 0XFFEF;break;
        case 5:ADPCFG = 0XFFDF;break;
        case 6:ADPCFG = 0XFFBF;break;
        case 7:ADPCFG = 0XFF7F;break;
        case 8:ADPCFG = 0XFEFF;break;
        case 9:ADPCFG = 0XFDFF;break;
        case 10:ADPCFG = 0XFBFF;break;
        case 11:ADPCFG = 0XF7FF;break;
        case 12:ADPCFG = 0XEFFF;break;
    }
    ADCON1 = 0;
    ADCON2 = 0;
    ADCON3 = 0x0E0D;
    ADCHS = channel;
    ADCSSL = 0;
    _ADON = 1;
    _SAMP = 1;
    delay_us(5);
    _SAMP = 0;
    while (_DONE == 0);
    return ADCBUF0;
}
```

Figure 4-6-2: ADC module handling program code

Line 2 to 16 of the code is used to indicate which pin to use as an analog input by clearing the related bit. For the case of sensor glove, channel 3, 4, 5, 6 and 7 will be selected at each time to sample the voltage value of the flexible sensor. ADCON1 is written with 0 to indicate off the ADC and disable the sampling process since we are still in the initial stage to initialize the ADC module, ADC continue operation when in IDLE mode, integer output, clear SAMP bit to stop sampling and start conversion and disable auto-START bit. While ADCON2 is also written with 0 values to indicate voltage reference to AVDD and AVSS and others bit are consider as don't care and therefore filled with 0. ADCON3 is written with 0x0E0D to indicate 14 TAD sample time, clock source from system clock and 7 x TCY (7 x 50 nanoseconds = 350 nanoseconds) of conversion time. ADCHS is written with channel number wanted to indicate which analog input been selected and ADCSSL is clear to indicate no pin scan for any analog input pin. After the configuration of the ADC module has complete, direct turn on the module and start sampling. Then create a system clock delay of 5 microseconds and wait for the sampling process to finish. After the sampling process finish, the sample value will be in the ADCBUF0.

Using the same method as discussed above to sample for the channel wanted and finally, all the sensor value of the sensor glove has been get. Next step to do is to calibrate the sensor glove in which we will set that when all the fingers in relax, the final data value of each finger will be 0 and when all the fingers are contract, the final data value of each finger will be 180. This is very useful since the length of each finger is different and thus the sample value will be different among fingers. By normalize all the sensor data of the sensor glove into a single standard, the range of sensor data for each finger will be limit in the range of 0 to 180.

4.7 Joystick programming

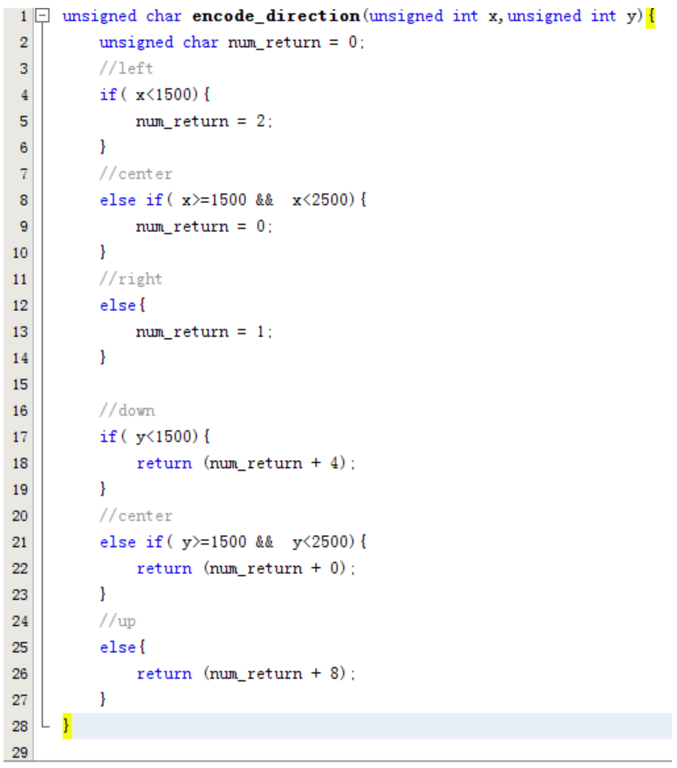
Refer to the circuit shown in section 3.6, now we will now proceed to the programming part. We will discuss about the data handling of the analog joystick part first then only discussed for the digital key press. For the analog joystick, there are 2 analog value that we need to handle, x-axis and y-axis. X-axis is control for the direction of left and right and y-axis is control for the direction of up and down. With using the same method that discuss in last section about using ADC module of the dsPIC30F4013 microcontroller to get the voltage value and transform it into digital data, the x-axis and y-axis data can be get using the same ways to and we will now focus on the data handling of it. Following table shows the encoded value encode from the direction of x-axis and y-axis.

Up	Down	Left	Right	Remark	Code
0	0	0	0	Center, not move	0x00
0	0	0	1	Right	0x01
0	0	1	0	Left	0x02
0	0	1	1	No exists, x-axis cannot move both direction simultaneously	0x03
0	1	0	0	Reverse	0x04
0	1	0	1	Right reverse	0x05
0	1	1	0	Left reverse	0x06
0	1	1	1	No exists, x-axis cannot move both direction simultaneously	0x07
1	0	0	0	Forward	0x08
1	0	0	1	Right forward	0x09
1	0	1	0	Left forward	0x0A
1	0	1	1	No exists, x-axis cannot move both direction simultaneously	0x0B
1	1	0	0	No exists, y-axis cannot move both direction simultaneously	0x0C
1	1	0	1	No exists, y-axis cannot move both direction simultaneously	0x0D
1	1	1	0	No exists, y-axis cannot move both direction simultaneously	0x0E

1	1	1	1	No exists, y-axis cannot move both direction simultaneously	0x0F

Table 4-7-1: Thumb joystick encoded command code

With referring to encoded code as shown in table above, diagram below shows the code of the data handling of the analog joystick.



```
unsigned char encode_direction(unsigned int x,unsigned int y){
    unsigned char num_return = 0;
    //left
    if( x<1500){
        num_return = 2;
    }
    //center
    else if( x>=1500 &&  x<2500){
        num_return = 0;
    }
    //right
    else{
        num_return = 1;
    }

    //down
    if( y<1500){
        return (num_return + 4);
    }
    //center
    else if( y>=1500 &&  y<2500){
        return (num_return + 0);
    }
    //up
    else{
        return (num_return + 8);
    }
}
```

Figure 4-7-1: Thumb joystick analog data handling

With the digital value of x-axis and y-axis pass into this function, this function will return an 8-bit number. Note that the x-axis and y-axis encode value only take place in the low nibble of the 8-bit encoded code, the high nibble will be used by the digital key press.

For the digital key, we will need to handle for the A, B, C, D and E keys while the F key is not used. Each key will represent an action and it cannot combine with another other digital key to perform an action. Following table shown how we encode the key press into an 8-bit number and note that the digital key presses only take place in the high nibble of the 8-bit number.

A	B	C	D	Remark	Code
0	0	0	0	No key press	0x00
0	0	0	1	Press D key	0x10
0	0	1	0	Press C key	0x20
0	0	1	1	Don’t care	0x30
0	1	0	0	Press B key	0x40
0	1	0	1	Don’t care	0x50
0	1	1	0	Don’t care	0x60
0	1	1	1	Don’t care	0x70
1	0	0	0	Press A key	0x80
1	0	0	1	Don’t care	0x90
1	0	1	0	Don’t care	0xA0
1	0	1	1	Don’t care	0xB0
1	1	0	0	Don’t care	0xC0
1	1	0	1	Don’t care	0xD0
1	1	1	0	Don’t care	0xE0
1	1	1	1	Don’t care	0xF0

Table 4-7-2: Joystick key encoded command code

Key E pressed will be representing with 0XFF since this number is unachievable for the combination of direction and digital key press. Then, when we combine both direction and digital key value, a series of action can be set.

Chapter 5 System Integration

5.1 Hardware Block Diagram

With the hardware and software parts been discussed in previous section, now we will need to combine all the hardware parts to become a fully functional mobile robot and all the software parts to become a robust system that is able to control the mobile robot precisely and accurately. To proceed, first thing to do is that define the block diagram of the mobile robot. The diagram below shows the block diagram of the mobile robot of this project.

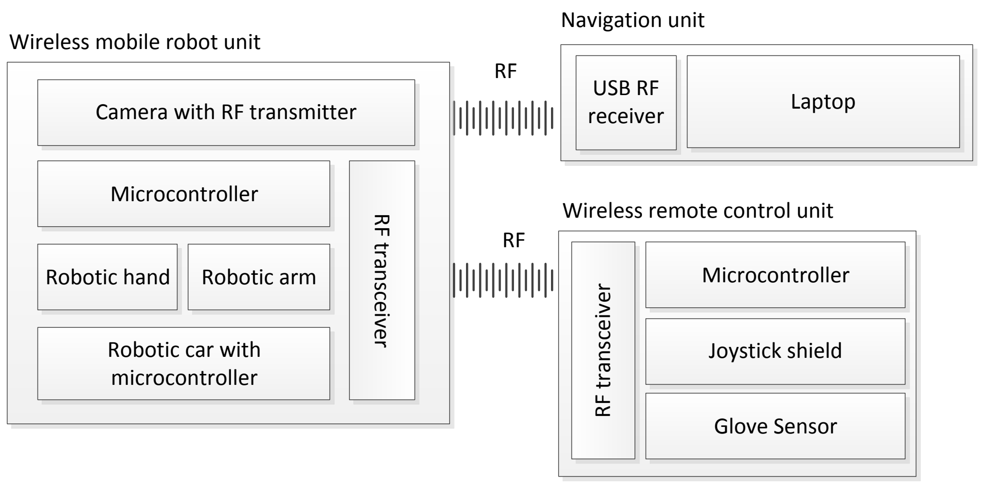


Figure 5-1-1: Full design block diagram

From the block diagram above, we can clearly investigate that there are 2 separate unit been partitioned which are wireless mobile robot unit and the wireless remote control unit. The integration of each unit will be discussed in later section.

5.2 Software Integration

5.2.1 Flow Chart and Encoded Command code

For the system flow, since we had use 3 microcontroller to integrate the hardware, thus we will had 3 sets of program code. However, in the sense of system design, we will only separate the whole system into 2 parts, which are transmitter and receiver. Transmitter system may refer to the wireless remote control unit in which the wireless remote control unit gather the information and send it to the receiver and no feedback needed. While for receiver system, it may refer to the wireless mobile robot unit since it is only receive the command and perform the action wanted. For the receiver system also, since there are 2 microcontroller used to handle all the operation of the mobile robot, extra communication is establish which is by UART. Since the RF transceiver is connected to the robotic car's microcontroller, thus the robotic car's microcontroller is responsible to transfer the data to another microcontroller which is the robotic hand's microcontroller. The diagrams below show the system flow chart of both the transmitter system and the receiver system and the table shows the encoded command code of both transmitter system and receiver system.

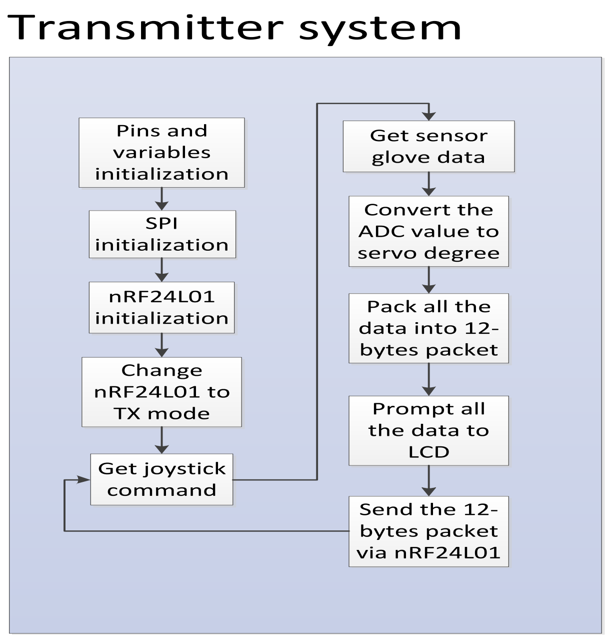

Transmitter system
Pins and variables initialization
SPI initialization
nRF24L01 initialization
Change nRF24L01 to TX mode
Get joystick command
Get sensor glove data
Convert the ADC value to servo degree
Pack all the data into 12-bytes packet
Prompt all the data to LCD
Send the 12-bytes packet via nRF24L01

Figure 5-2-1: Transmitter system program flow

Code (decimal)	Remark
0	Not pressing any key
1	Thumb joystick to pull right direction
2	Thumb joystick to pull left direction
4	Thumb joystick to pull down direction
5	Thumb joystick to pull down right direction
6	Thumb joystick to pull down left direction
8	Thumb joystick to pull up direction
9	Thumb joystick to pull up right direction
10	Thumb joystick to pull up left direction
17	Press key D + Thumb joystick to pull right direction
18	Press key D + Thumb joystick to pull left direction
33	Press key C + Thumb joystick to pull right direction
34	Press key C + Thumb joystick to pull left direction
65	Press key B + Thumb joystick to pull right direction
66	Press key B + Thumb joystick to pull left direction
129	Press key A + Thumb joystick to pull right direction
130	Press key A + Thumb joystick to pull left direction
132	Press key A + Thumb joystick to pull down direction
136	Press key A + Thumb joystick to pull up direction
255	Press key E

Table 5-2-1-1: Transmitter system encoded command code

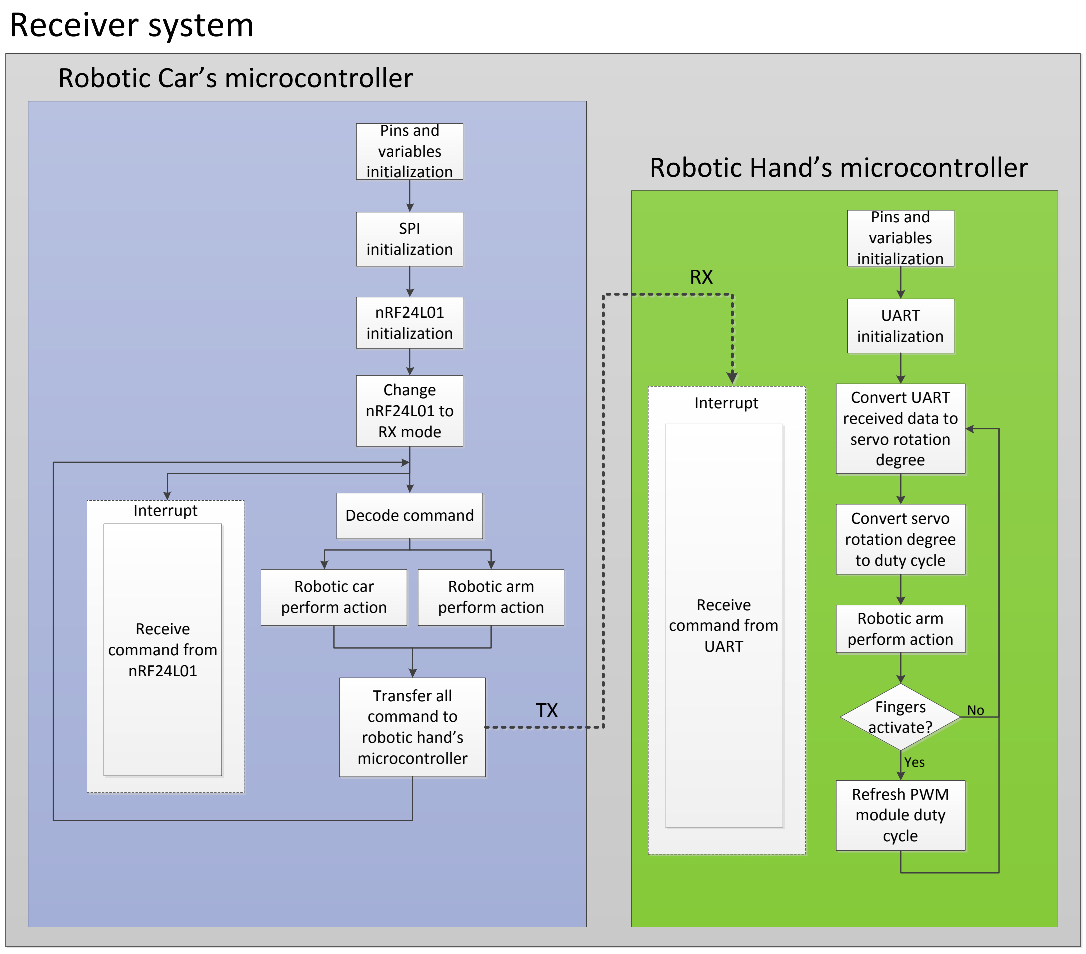

Receiver system
Robotic Car's microcontroller
Pins and variables initialization
SPI initialization
nRF24L01 initialization
Change nRF24L01 to RX mode
Interrupt
Receive command from nRF24L01
Decode command
Robotic car perform action
Robotic arm perform action
Transfer all command to robotic hand's microcontroller
TX
RX
Robotic Hand's microcontroller
Pins and variables initialization
UART initialization
Interrupt
Receive command from UART
Convert UART received data to servo rotation degree
Convert servo rotation degree to duty cycle
Robotic arm perform action
Fingers activate?
No
Yes
Refresh PWM module duty cycle

Figure 5-2-2: Receiver system program flow

Code (decimal)	Microcontroller in-charge	Remark
0	Robotic car	Robotic car stop moving
1	Robotic car	Robotic car turns right
2	Robotic car	Robotic car turns left
4	Robotic car	Robotic car moving reverse
5	Robotic car	Robotic car moving reverse and right turn
6	Robotic car	Robotic car moving reverse and left turn
8	Robotic car	Robotic car moving forward
9	Robotic car	Robotic car moving forward and right turn

10	Robotic car	Robotic car moving forward and left turn
17	Robotic hand	wrist_servo_1 rotate right
18	Robotic hand	wrist_servo_1 rotate left
33	Robotic car	wrist_servo_2 rotate right
34	Robotic car	wrist_servo_2 rotate left
65	Robotic car	elbow_servo rotate right
66	Robotic car	elbow_servo rotate left
129	Robotic car	shouder_servo rotate right
130	Robotic car	shouder_servo rotate left
132	Robotic car	base_servo rotate down
136	Robotic car	base_servo rotate up
255	Robotic hand	Activate / deactivate robotic hand's servos to rotate

Table 5-2-2: Receiver system encoded command code

5.2.2 Wireless Packet Transmission Protocol

For the data send by wireless remote control unit, there are total of 12 bytes data being send. This 12 bytes data consist of 2 bytes for joystick encoded command code and 10 bytes for sensor glove data. First 4 bits of every byte are used to define which data has been received from the receiver side. Table below shows the function table of the 4 bits value stated.

Value (4-bits)	Data stored
0000	Lower nibble of the joystick encoded command code
0001	Upper nibble of the joystick encoded command code
0010	Lower nibble of the first finger
0011	Upper nibble of the first finger
0100	Lower nibble of the second finger
0101	Upper nibble of the second finger
0110	Lower nibble of the third finger
0111	Upper nibble of the third finger
1000	Lower nibble of the forth finger
1001	Upper nibble of the forth finger
1010	Lower nibble of the fifth finger
1011	Upper nibble of the fifth finger
1100	Don’t care
1101	Don’t care
1110	Don’t care
1111	Don’t care

Table 5-2-3: Packet structure table

For transmitter side, the lower nibble is process by perform bitwise AND with 0x0F then bitwise OR with the 4 bits value of respective data set while for upper nibble is process by perform shifting 4 bits towards right then bitwise OR with the 4 bits value of respective data set.

For receiver side, in order to allow fast response when a data has been transmit to the receiver side, an external interrupt pin of the dsPIC30F4013 microcontroller, RD9 is connected to IRQ pin of the nRF24L01 transceiver module. When the data reach the nRF24L01 transceiver module, nRF24L01 will pull low IRQ pin for 1 cycle. Thus, we will need to capture this falling edge and collect the data when falling edge is captured. Diagram below shows the timing diagram of respective condition.

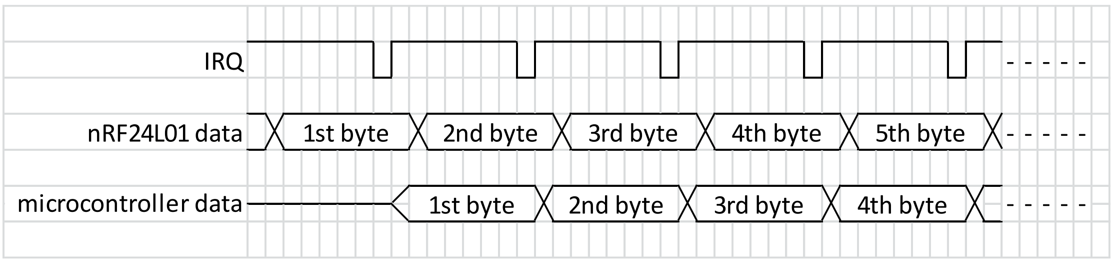
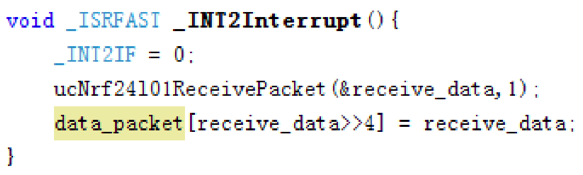


Figure 5-2-3: Data receiving timing requirement

With refers to timing diagram, whole packet can be received without any collapse between data. Following shows the code used to handle the data receive in receiver side.



```
void _ISRFAST _INT2Interrupt(){
    _INT2IF = 0;
    ucNrf24l01ReceivePacket(&receive_data,1);
    data_packet[receive_data>>4] = receive_data;
}
```

Figure 5-2-4: Data receiving ISR handling code

First 4 bits of each byte in the receiver side are treated as don’t care and can be obsolete by perform bitwise AND with 0x0F.

5.3 Hardware Integration

5.3.1 Wireless mobile robot unit

With the full hardware block diagram shown in section 5-1, in this section, we will focus on the integration of wireless mobile robot unit. The wireless mobile robot unit consists of a robotic car as moving based, a robotic arm as to control the position of hand, a robotic hand to grasp and hold an object and a RF transceiver as a bridge of connection with another pair of RF transceiver wirelessly. There are 2 sets of dsPIC30F4013 microcontroller used by this unit because this unit needs at least 12 PWM modules and since 1 microcontroller can only produce maximum up to 7 PWM output, thus 2 sets of dsPIC30F4013 microcontrollers are required. Following shows the full circuit diagram of the wireless mobile robot unit of this project and the table shows the command code for the action of the mobile robot.

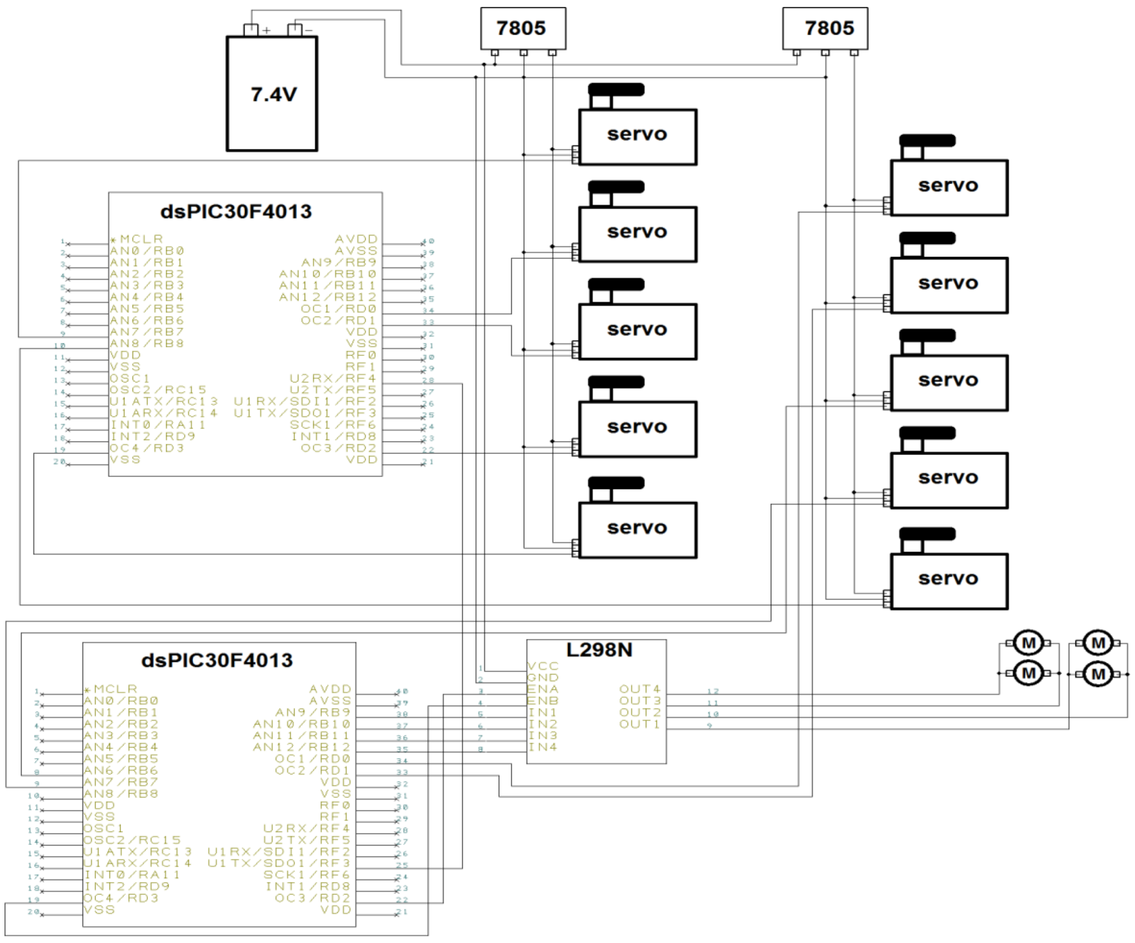


Figure 5-3-1: Full schematic view of wireless mobile robot unit

5.3.2 Wireless remote control unit

For the wireless remote control unit, it is used to send the command of user to the mobile robot to perform an action. This unit consists of a joystick shield used to control the action of the robotic car and robotic arm, a sensor glove used to control the motions of the robotic hand and a RF transceiver used to transmit the command to another pair of RF transceiver. Following shows the full circuit diagram of the wireless remote control unit of this project and the command code encoded from the joystick.

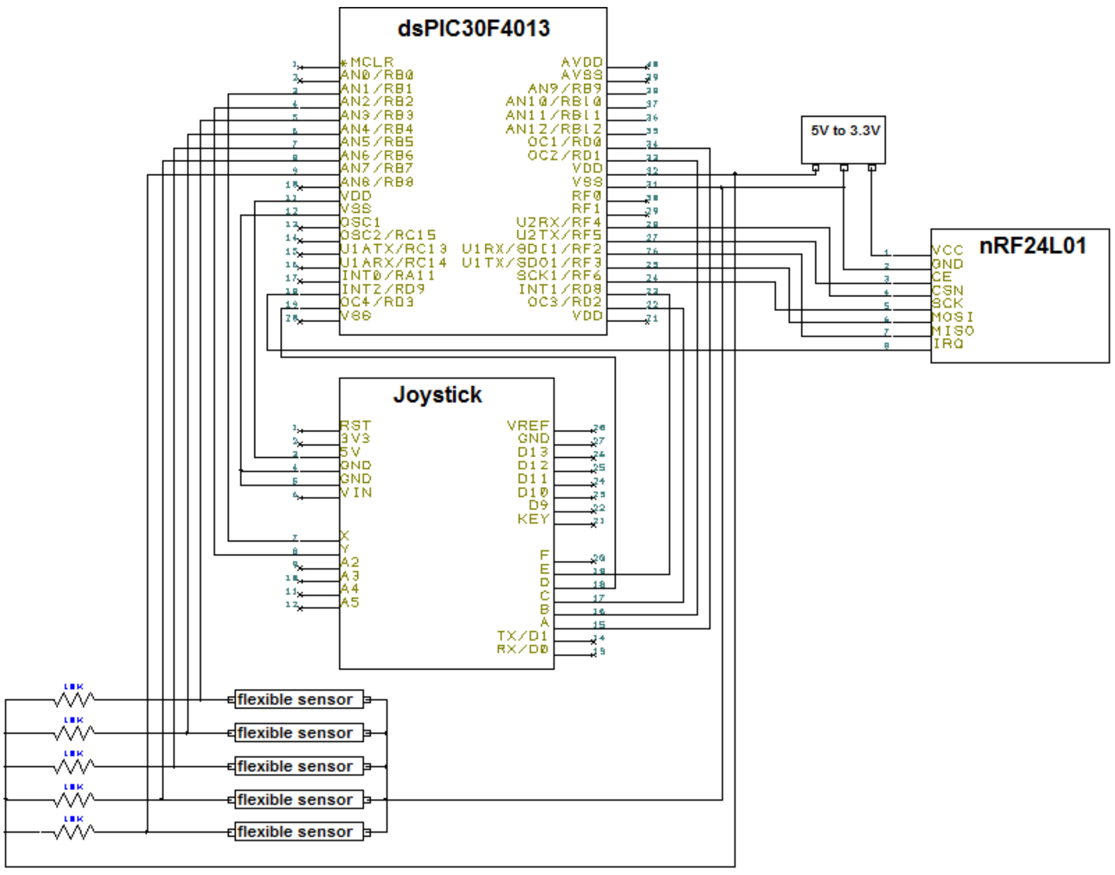


Figure 5-3-2: Full schematic view of wireless remote control unit

5.4 Final Product Review

After done for hardware and software integration, the final product has finally successfully developed. Following diagrams show the wireless mobile robot unit and the wireless remote control unit.

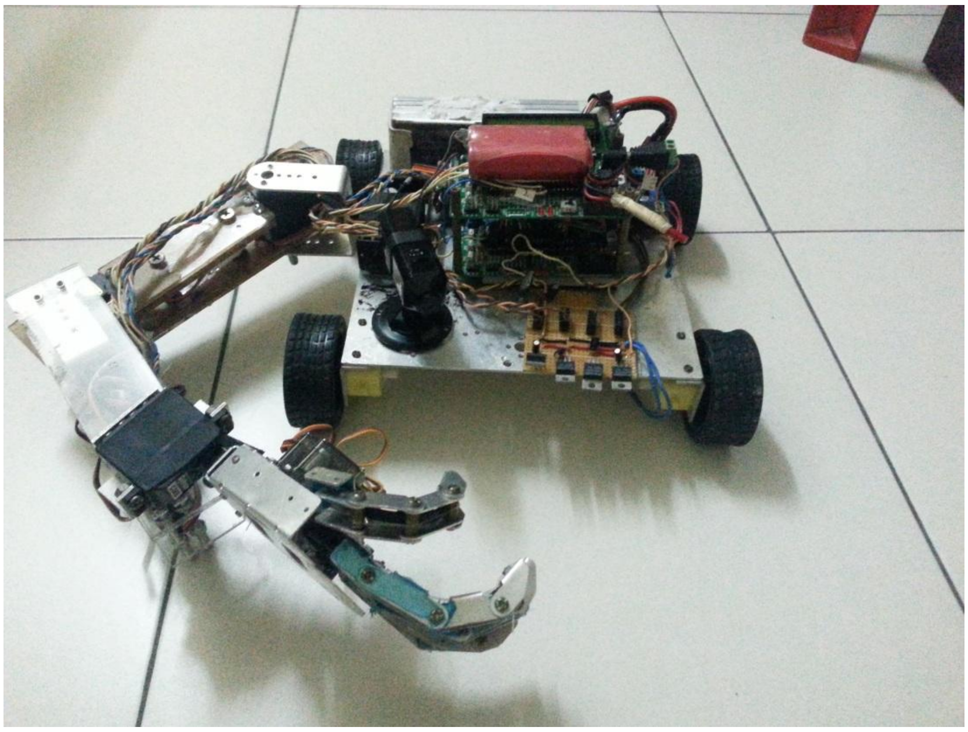


Figure 5-4-1: Mobile robot unit front view

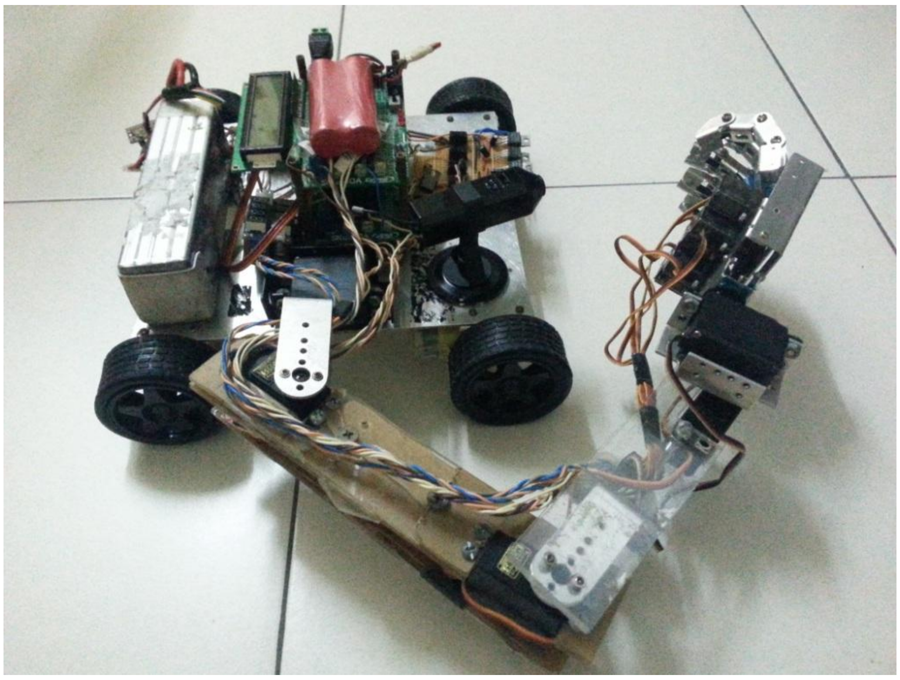


Figure 5-4-2: Mobile robot unit side view

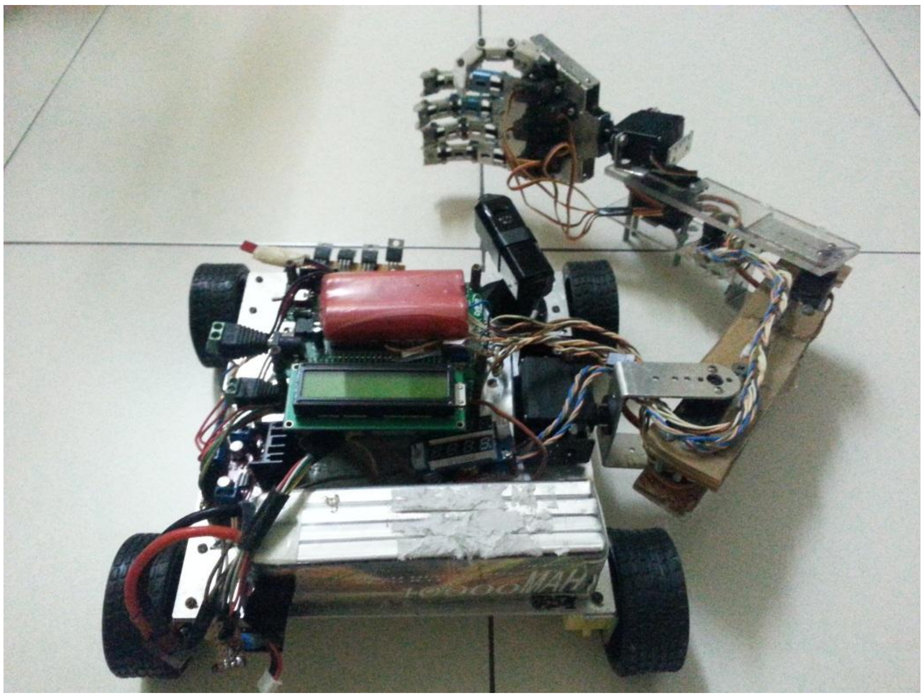


Figure 5-4-3: Mobile robot unit rear view

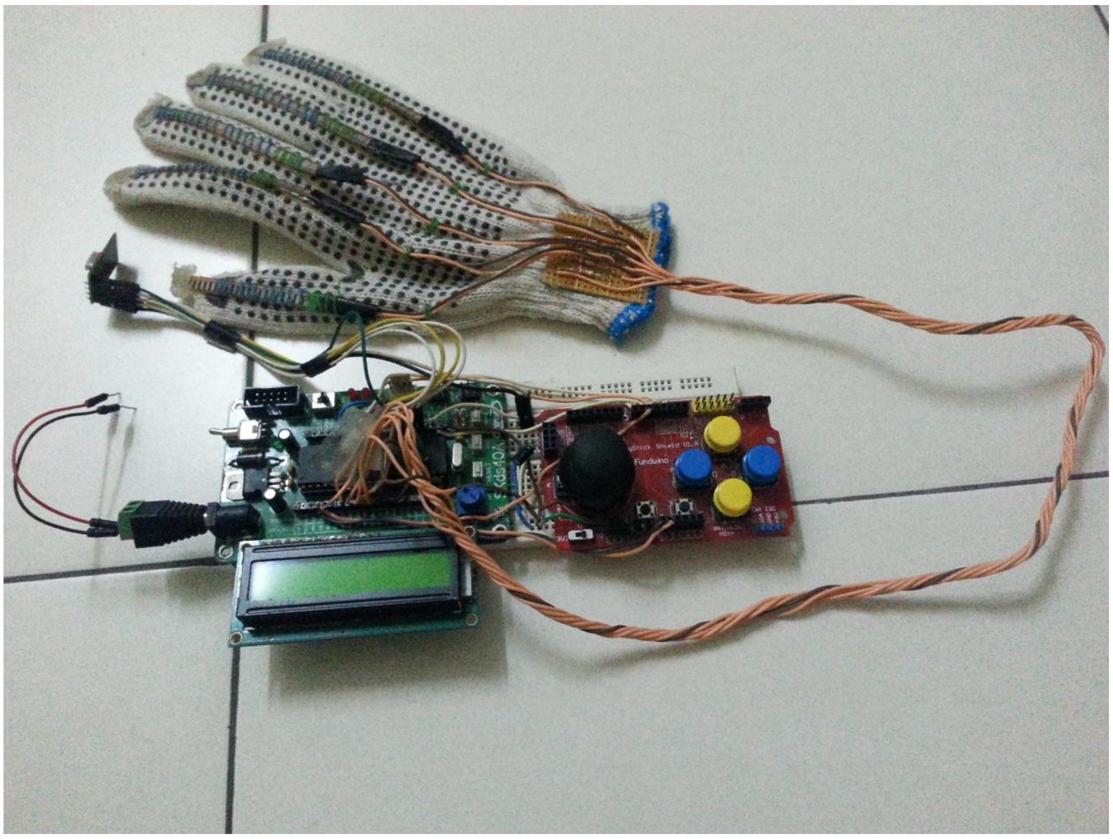


Figure 5-4-4: Wireless control unit with sensor glove

Chapter 6 Robot Application Development

6.1 Fire Detection

6.1.1 Tool and equipment

After the full robot had been completely constructed, now we will build an application on top of the resources use by the mobile robot. Note that from the block diagram shown in section 5.1, we can investigate that there is a camera put on the mobile robot unit. This camera is used to send the real time video capture to the user's laptop so that to allow user to navigate the mobile robot to a place that potentially harm to human body, which consider as a dangerous task. The application that we make is a fire detection system. This application used the video capture by the camera, perform the image processing and prompt user when detected a fire. The programming code of this application is written using C++ language. To allow performing image processing in C++ language, extra add-on need to be added which is the OpenCV library. For the wireless camera part, it is separate into 2 parts, the camera with RF transmitter and the USB RF receiver. There is nothing for camera with RF transmitter to do except capturing the video and send it to the USB receiver. For the USB receiver, since it is a standard Windows driver, the driver file need to be install first in order to allow the USB receiver to receive the video capture by the camera. After install the driver file, an extra software is needed to convert the DVcam video format to a normal video format which by DVdriver. Without this software, no video will be loaded since the existing format is incompatible.

By now, we should have no obstacles to begin our development on the robot application. Below show the images of the wireless camera, USB receiver, DVdriver software and the IDE used for develop the application which is the Microsoft Visual Studio 2010.

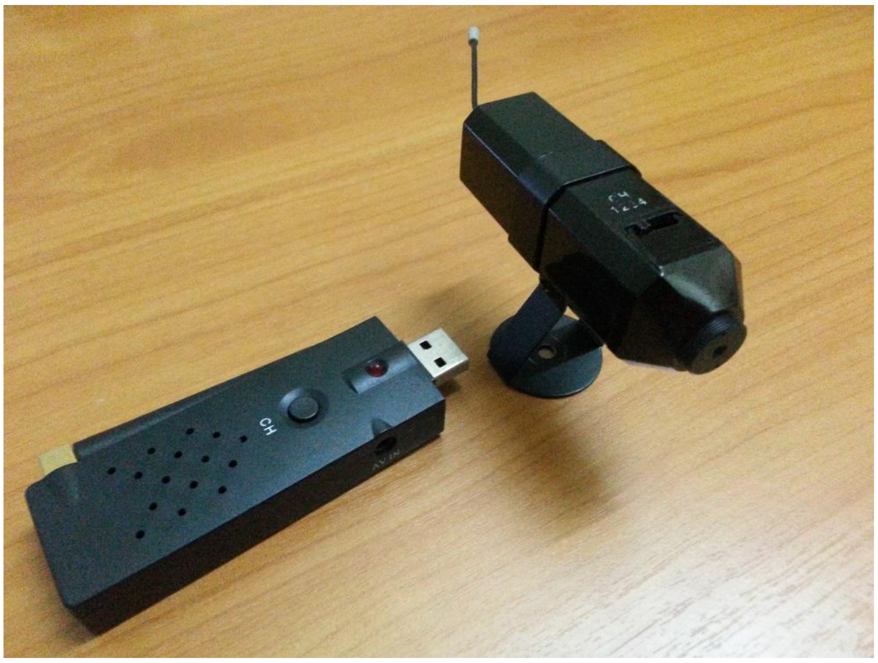
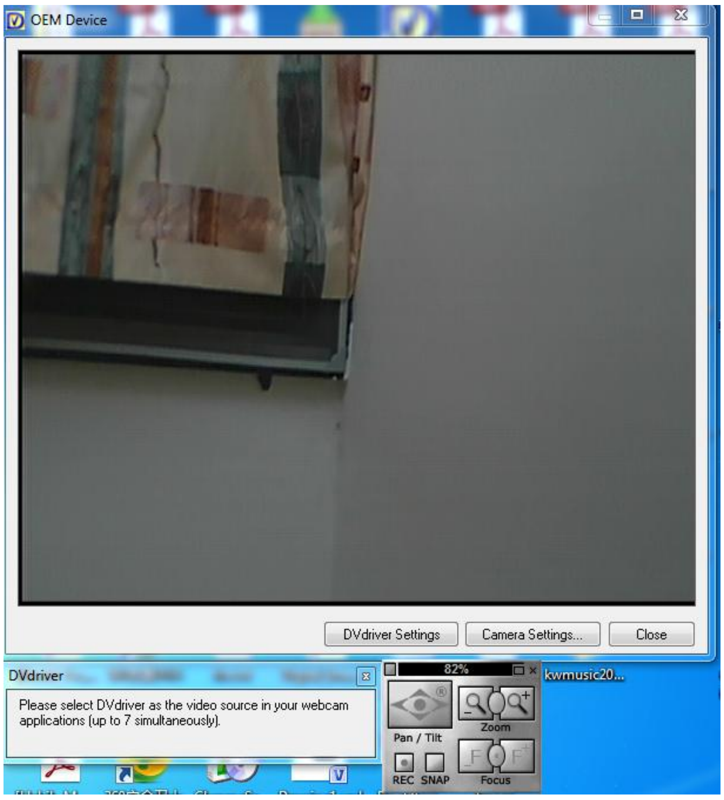
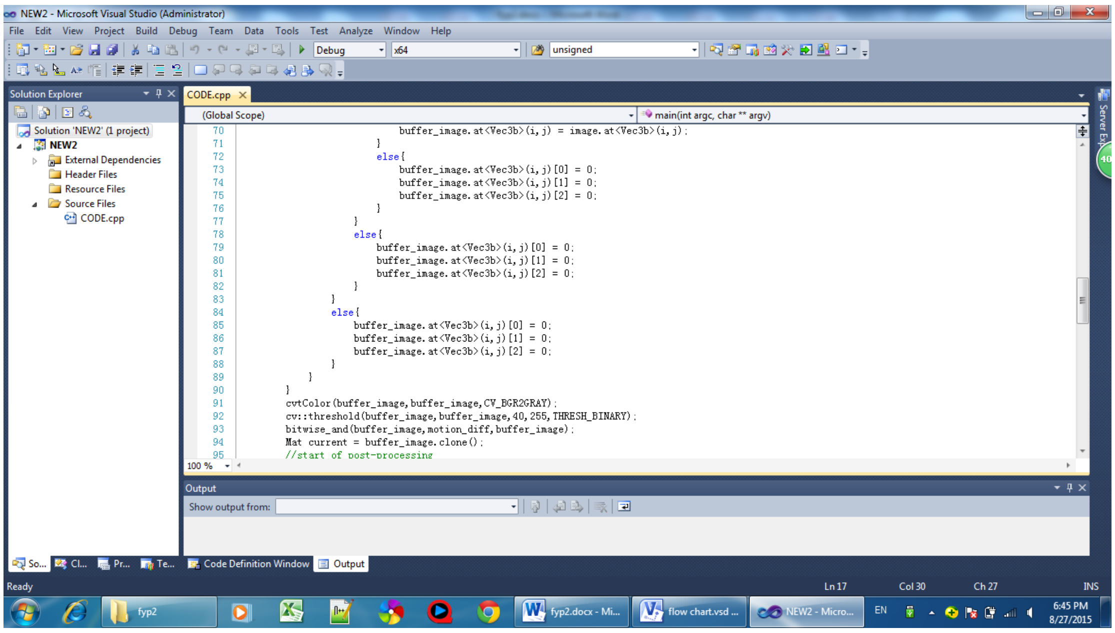


Figure 6-1-1: Wireless camera and USB receiver



Please select DVdriver as the video source in your webcam applications (up to 7 simultaneously).

Figure 6-1-2: DVdriver software



```
                        buffer_image.at<Vec3b>(i,j) = image.at<Vec3b>(i,j);
                    }
                    else{
                        buffer_image.at<Vec3b>(i,j)[0] = 0;
                        buffer_image.at<Vec3b>(i,j)[1] = 0;
                        buffer_image.at<Vec3b>(i,j)[2] = 0;
                    }
                }
                else{
                    buffer_image.at<Vec3b>(i,j)[0] = 0;
                    buffer_image.at<Vec3b>(i,j)[1] = 0;
                    buffer_image.at<Vec3b>(i,j)[2] = 0;
                }
            }
            else{
                buffer_image.at<Vec3b>(i,j)[0] = 0;
                buffer_image.at<Vec3b>(i,j)[1] = 0;
                buffer_image.at<Vec3b>(i,j)[2] = 0;
            }
        }
    }
    cvtColor(buffer_image,buffer_image,CV_BGR2GRAY);
    cv::threshold(buffer_image,buffer_image,40,255,THRESH_BINARY);
    bitwise_and(buffer_image,motion_diff,buffer_image);
    Mat current = buffer_image.clone();
    //start of post-processing
```

Figure 6-1-3: Microsoft Visual Studio 2010

6.1.2 Algorithm and flow chart

After setup the development tools and environment, now we will proceed to the stage of develop the application algorithm and program flow. To detect a fire, we will start with learning the characteristic of the fire. Due to there are a variety of technique to detect the fire, we will only specific the image processing technique using RGB color model image and in spatial domain.

As in RGB color model, a color image consists of 3 layer of colors, which are red, green and blue or in short RGB. Each of this color contain a value that represent the intensity of the respective layer of color, in which the lower the value the darkest, the higher the value the brightest. In order to detect the fire, based on some existing research that has been done, 3 sets of rules been define, which red color intensity must more than an offset value, red color intensity is greater than green color intensity and green color intensity must greater than blue color intensity. With using this method, majority of the fire image can be detected. However, it is not efficient if there is an item that looks similar in color with fire.

Due to this reason, the second layer of detection has been developed, the motion detection of fire. Since we are using video capture by camera and doing the image processing based on the video, a video may have several images collapse sequentially to become a video. For example, in a 1 second length of video may have contained more than 20 images collapse sequentially. Let (n) be the current image, we may use the past image, (n-1) image and the current image, (n) image to do a comparison. With this, the motion of the fire can be clear detected since the flame always moving from time to time.

Up to now, the fire detection algorithm has been fully define and following diagram shows the flow chart of the algorithm been discussed.

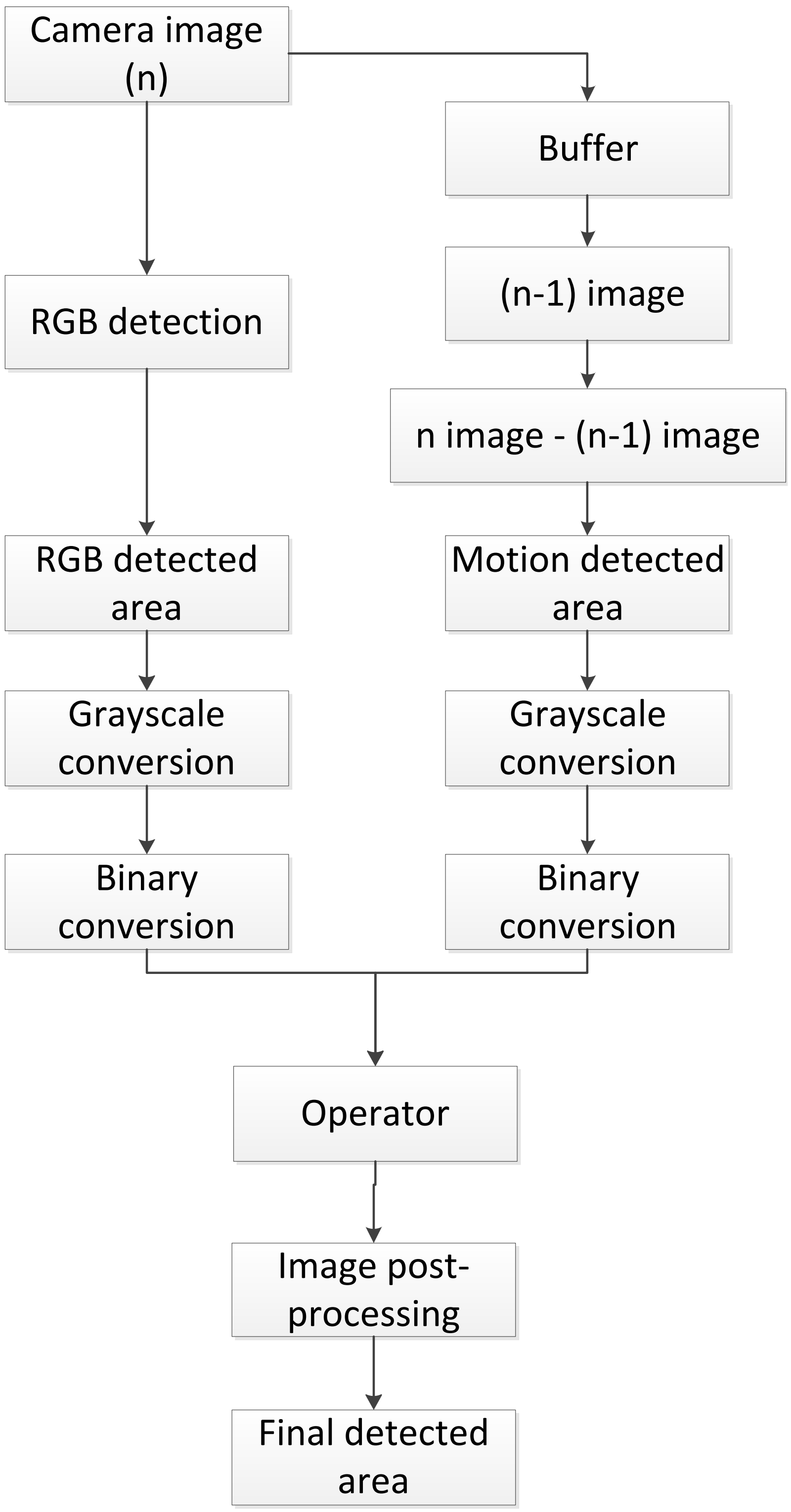


Figure 6-1-4: Fire detection algorithm flow

6.1.3 Implementation

6.1.3.1 Fire Detecting

With refers to the algorithm that describe, now we will need to realize it into program code. The whole program code can be divided into 3 parts which are image acquisition, image processing and image recognition. First 2 parts stated are under fire detecting categories while the image recognition is under the next section - Bounding Box on fire and will be discuss later.

By now, all the driver files and the software of the application development should be ready and now we will precede our focus into the image acquisition part. Image acquisition simply means that acquiring the image and few lines of code has been written. Following codes shows how we proceed for the image acquisition.

```
Mat image;
VideoCapture cap;
cap.open(cam_num);
cap>>image;
```

The codes shown above will handles for the initialization of the variable, open and read the camera driver, acquire the image and store it into a 2-d matrix data type.

While for the image processing part, in which we will realize the algorithm that discuss previous into the program code. According to the algorithm discussed, there are 2 layers of detection used, which are RGB detection and motion detection. We will focus on the motion detection first then only perform color detection. For the motion detection, it is just some simple code that perform absolute subtraction between previous image, (n-1) and current image, (n). Following lines of code show the motion detection algorithm with previous image denoted as buffer_image and current image denoted as image, the conversion of the resultant image to the grayscale image and the conversion of the grayscale image to binary image.

```
absdiff(buffer_image, image, motion_diff);
cvtColor(motion_diff,motion_diff,CV_BGR2GRAY);
cv::threshold(motion_diff,motion_diff,40,255,THRESH_BINARY);
```

As describe in previous, the RGB detection define 3 rules that shown below.

1) Red color intensity > offset value
2) Red color intensity > green color intensity
3) Green color intensity > blue color intensity

These 3 rules will be set as condition in the program code, after perform the color detection, convert the result image from RGB color space to grayscale color space and finally convert the grayscale image to binary image. The code for the color detection and the conversion to binary image are shown below.

```
#pragma omp parallel for
for(int i=image.rows-1;i>=0;i--){
for(int j=image.cols-1;j>=0;j--){
        if(image.at<Vec3b>(i,j)[2]>threshold) {
        if(image.at<Vec3b>(i,j)[2]>image.at<Vec3b>(i,j)[1]){
        if(image.at<Vec3b>(i,j)[1]>image.at<Vec3b>(i,j)[0]){
                buffer_image.at<Vec3b>(i,j)[0] = image.at<Vec3b>(i,j)[0];
                buffer_image.at<Vec3b>(i,j)[1] = image.at<Vec3b>(i,j)[1];
                buffer_image.at<Vec3b>(i,j)[2] = image.at<Vec3b>(i,j)[2];
                }else{
                        buffer_image.at<Vec3b>(i,j)[0] = 0;
                        buffer_image.at<Vec3b>(i,j)[1] = 0;
                        buffer_image.at<Vec3b>(i,j)[2] = 0;
                }}else{
                        buffer_image.at<Vec3b>(i,j)[0] = 0;
                        buffer_image.at<Vec3b>(i,j)[1] = 0;
                        buffer_image.at<Vec3b>(i,j)[2] = 0;
                }
        }else{
                buffer_image.at<Vec3b>(i,j)[0] = 0;
                buffer_image.at<Vec3b>(i,j)[1] = 0;
                buffer_image.at<Vec3b>(i,j)[2] = 0;
        }}}
cvtColor(buffer_image,buffer_image,CV_BGR2GRAY);
cv::threshold(buffer_image,buffer_image,40,255,THRESH_BINARY);
```

Since the image acquired is in spatial domain, hence the image processing will be process pixel by pixel. The image acquired is in BGR sorting, thus image.at<Vec3b>(i,j)[2] will represent for red color layer, image.at<Vec3b>(i,j)[1] will represent for green color layer and image.at<Vec3b>(i,j)[0] will represent for blue color layer. Threshold variable inside the code is used to store the offset value, which is stated in the first rule of the RGB detection and it is set by user. When a pixel met all the condition, it will be proceed with next level processing while the pixel that does not met all the condition will be transform to black color pixel.

With getting the binary image of both color detection and motion detection, now perform bitwise AND operation and flame is detected. The result of the application is as shown in diagram below.

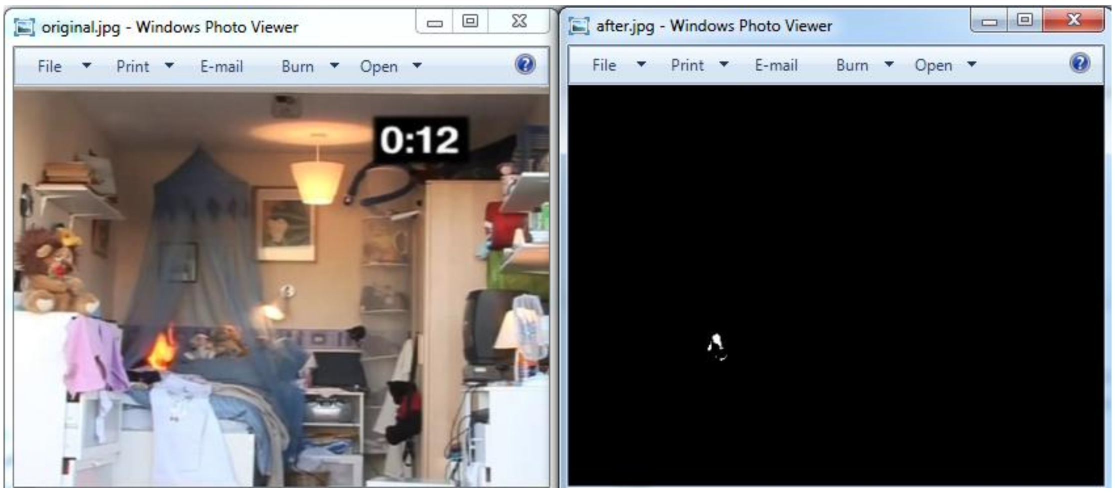

original.jpg - Windows Photo Viewer
File
Print
E-mail
Burn
Open
0:12
after.jpg - Windows Photo Viewer
File
Print
E-mail
Burn
Open

Figure 6-1-5: Initial result of fire detection application

From the diagram, the fire detected is not very obvious to be seen and thus, extra image post-processing needed to take out. Following diagram shows the post-processing program code that used to enhance the detected area.

```
erode(buffer_image, buffer_image, getStructuringElement(MORPH_ELLIPSE, Size(2,2)));
dilate(buffer_image, buffer_image,getStructuringElement(MORPH_ELLIPSE, Size(8,8)));
blur(buffer_image,buffer_image,Size(2,2));
dilate(buffer_image, buffer_image,getStructuringElement(MORPH_ELLIPSE, Size(12,12)));
```

The first line of code used to remove to small object that appear in the image or it would be called noise to the image while the second line of the code perform to fill the small holes after all the noise been remove. To get the better resulting image, third line to end of the code is used to further enhance the fire detected area and following diagram shows the image of fire been detected after a series of post-processing method.

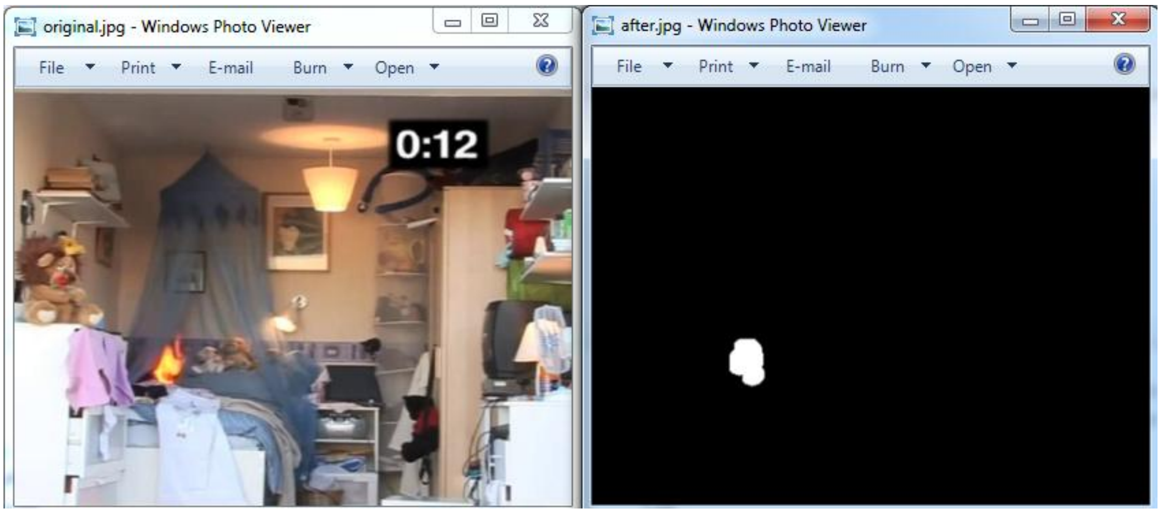
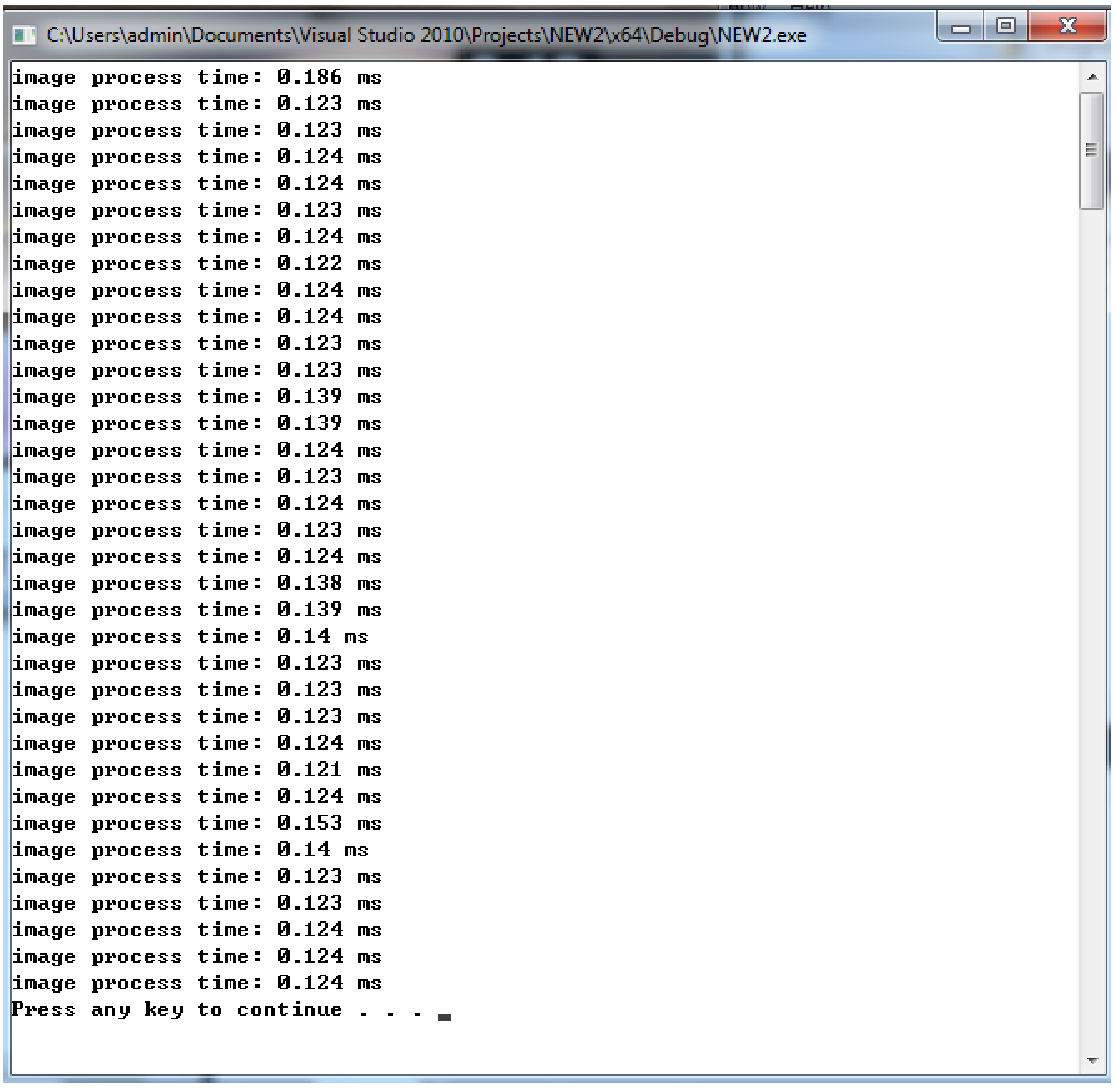

original.jpg - Windows Photo Viewer
File
Print
E-mail
Burn
Open
0:12
after.jpg - Windows Photo Viewer
File
Print
E-mail
Burn
Open

Figure 6-1-6: Final result of fire detection application

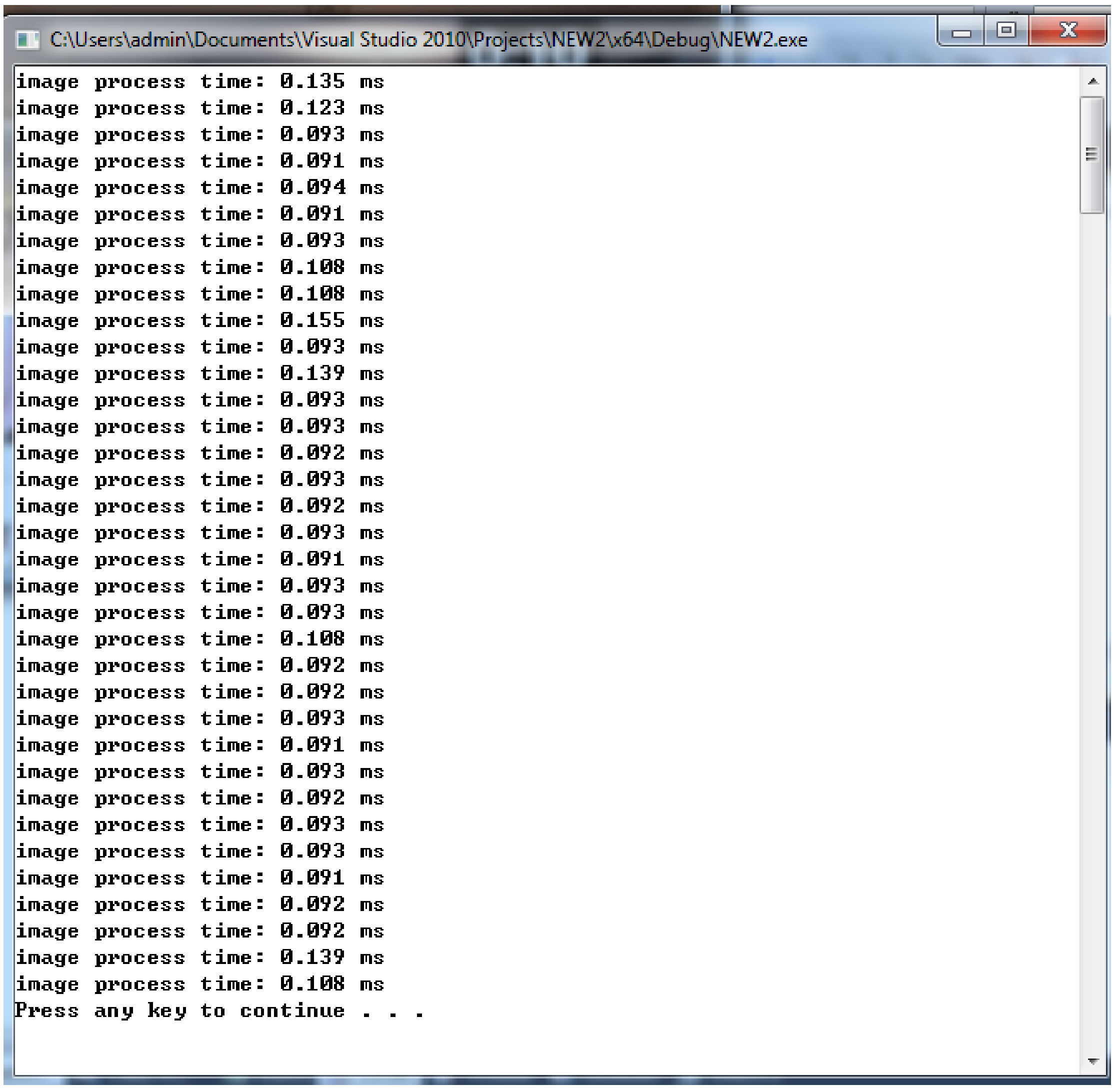
However, in the view of performance, the image processing time is considered quite slow even with the maximum of optimization. The diagram below shows the image processing time of current optimization approach.


C:\Users\admin\Documents\Visual Studio 2010\Projects\NEW2\x64\Debug\NEW2.exe

```
image process time: 0.186 ms
image process time: 0.123 ms
image process time: 0.123 ms
image process time: 0.124 ms
image process time: 0.124 ms
image process time: 0.123 ms
image process time: 0.124 ms
image process time: 0.122 ms
image process time: 0.124 ms
image process time: 0.124 ms
image process time: 0.123 ms
image process time: 0.123 ms
image process time: 0.139 ms
image process time: 0.139 ms
image process time: 0.124 ms
image process time: 0.123 ms
image process time: 0.124 ms
image process time: 0.123 ms
image process time: 0.124 ms
image process time: 0.138 ms
image process time: 0.139 ms
image process time: 0.14 ms
image process time: 0.123 ms
image process time: 0.123 ms
image process time: 0.123 ms
image process time: 0.124 ms
image process time: 0.121 ms
image process time: 0.124 ms
image process time: 0.153 ms
image process time: 0.14 ms
image process time: 0.123 ms
image process time: 0.123 ms
image process time: 0.124 ms
image process time: 0.124 ms
image process time: 0.124 ms
Press any key to continue . . .
```

Figure 6-1-7: Initial processing time

An approach has been done by using multiple threads to perform the execution. Since the image is process in spatial domain, each pixel is not depending on each other and no dependencies among the pixels. Thus, we had used openMP method by simply adding a line of code above the first for loop of the code shown in previous which is #pragma omp parallel for. Following diagram shows the result of using multiple threads to improve image processing time.



C:\Users\admin\Documents\Visual Studio 2010\Projects\NEW2\x64\Debug\NEW2.exe

```
image process time: 0.135 ms
image process time: 0.123 ms
image process time: 0.093 ms
image process time: 0.091 ms
image process time: 0.094 ms
image process time: 0.091 ms
image process time: 0.093 ms
image process time: 0.108 ms
image process time: 0.108 ms
image process time: 0.155 ms
image process time: 0.093 ms
image process time: 0.139 ms
image process time: 0.093 ms
image process time: 0.093 ms
image process time: 0.092 ms
image process time: 0.093 ms
image process time: 0.092 ms
image process time: 0.093 ms
image process time: 0.091 ms
image process time: 0.093 ms
image process time: 0.093 ms
image process time: 0.108 ms
image process time: 0.092 ms
image process time: 0.092 ms
image process time: 0.093 ms
image process time: 0.091 ms
image process time: 0.093 ms
image process time: 0.092 ms
image process time: 0.093 ms
image process time: 0.093 ms
image process time: 0.091 ms
image process time: 0.092 ms
image process time: 0.092 ms
image process time: 0.139 ms
image process time: 0.108 ms
Press any key to continue . . .
```

Figure 6-1-8: Final processing time

Based on both diagram, the performance has improved. For now, the fire detection algorithm has been fully processed. The later work to do is to put a rectangle on the area of the detected flame in which will be discussed in the next section.

6.1.3.2 Bounding Box on fire

After perform the fire detection as in the previous section, now we will focus on putting a rectangle on the flame area. It is pretty easy to code since openCV also provides APIs for detecting shape and drawing. With the image that fully process in the last section, the shape of the fire can easily been detected by using the code below.

```
vector< vector<Point> > contours;
findContours(buffer_image,contours,
CV_RETR_TREE ,CV_CHAIN_APPROX_SIMPLE);
```

This line of code used to find the contour of all the shape and store the information of the shape into a vector. The vector will store all the starting x and starting y of the shape that exists in the image. The only thing that we concern is that the flame area, thus we set that the biggest shape detected is the flame area and following codes are the implementation of detecting the shape, find the largest shape and draw rectangle on the largest shape.

```
findContours(buffer_image,contours,
CV_RETR_TREE ,CV_CHAIN_APPROX_SIMPLE);
boundRect.resize(contours.size());
areas.resize(contours.size());
minMaxLoc(Mat(areas),0,&max,0,&maxPosition);
if (contours.size() >= 1){
        rect_box = boundingRect(contours[maxPosition.y]);
        rectangle(image2, rect_box.tl(),rect_box.br(), CV_RGB(255, 0, 0), 3, 5, 0);
}
```

With the code shown above, the fire detection application has fully developed.

Chapter 7 Final Testing and Result Discussion

7.1 Robot testing

7.1.1 Result and discussion

In order to ensure the robot perform well and ready for dangerous task, several test case has been define and following table shows the result of each test case.

No	Test case	Result & discussion
1	Power on both mobile robot and remote control unit	- mobile robot start normally, robotic car stay still and not moving, robotic arm and hand moving to preset position - remote control unit start normally, all the sensor value shown in LCD
2	Push thumb joystick to up	- robotic car moving forward, robotic arm and hand stay still - remote control unit's LCD shown number 8 (indicate moving forward)
3	Push thumb joystick to down	- robotic car moving backward, robotic arm and hand stay still - remote control unit's LCD shown number 4 (indicate moving backward)
4	Push thumb joystick to left	- robotic car turns left, robotic arm and hand stay still - remote control unit's LCD shown number 2 (indicate turn to left)
5	Push thumb joystick to right	- robotic car turns right, robotic arm and hand stay still - remote control unit's LCD shown number 1 (indicate turn to right)
6	Wear the sensor glove and try to perform some hand	- robotic car, arm and hand stay still (because the robotic hand is not activate yet)

	gesture	- remote control unit's LCD keep updating the glove sensor value
7	Press button E and perform hand gesture using sensor glove	- robotic car and arm stay still, robotic hand action change as control using sensor glove - remote control unit's LCD keep updating the glove sensor value
8	Press button E again and observe the robotic hand	- robotic car and arm stay still, robotic hand action fixed with last hand gesture perform - remote control unit's LCD keep updating the glove sensor value
9	Press button A and observe the mobile robot	- robotic car, arm and hand stay still - remote control unit's LCD shown number 128 (indicate button A is pressed)
10	Press button B and observe the mobile robot	- robotic car, arm and hand stay still - remote control unit's LCD shown number 64 (indicate button B is pressed)
11	Press button C and observe the mobile robot	- robotic car, arm and hand stay still - remote control unit's LCD shown number 32 (indicate button C is pressed)
12	Press button D and observe the mobile robot	- robotic car, arm and hand stay still - remote control unit's LCD shown number 16 (indicate button D is pressed)
13	Press button A and push thumb joystick to up	- robotic car and hand stay still, the base servo of the robotic arm moving upward - remote control unit's LCD shown number (indicate robotic arm's base move upward)
14	Press button A and push thumb joystick to down	- robotic car and hand stay still, the base servo of the robotic arm moving downward - remote control unit's LCD shown number (indicate robotic arm's base move downward)
15	Press button A and push	- robotic car and hand stay still, the shoulder servo

	thumb joystick to left	of the robotic arm moving left - remote control unit's LCD shown number (indicate robotic arm's shoulder move left)
16	Press button A and push thumb joystick to right	- robotic car and hand stay still, the shoulder servo of the robotic arm moving right - remote control unit's LCD shown number (indicate robotic arm's shoulder move right)
17	Press button B and push thumb joystick to up	- robotic car, arm and hand stay still (robotic arm's elbow not program to move up and down) - remote control unit's LCD shown number (this number treated as don't care)
18	Press button B and push thumb joystick to down	- robotic car, arm and hand stay still (robotic arm's elbow not program to move up and down) - remote control unit's LCD shown number (this number treated as don't care)
19	Press button B and push thumb joystick to left	- robotic car and hand stay still, the elbow servo of the robotic arm moving left - remote control unit's LCD shown number (indicate robotic arm's elbow move left)
20	Press button B and push thumb joystick to right	- robotic car and hand stay still, the elbow servo of the robotic arm moving right - remote control unit's LCD shown number (indicate robotic arm's elbow move right)
21	Press button C and push thumb joystick to up	- robotic car, arm and hand stay still (robotic arm's wrist not program to move up and down) - remote control unit's LCD shown number (this number treated as don't care)
22	Press button C and push thumb joystick to down	- robotic car, arm and hand stay still (robotic arm's wrist not program to move up and down) - remote control unit's LCD shown number (this number treated as don't care)

23	Press button C and push thumb joystick to left	- robotic car and hand stay still, the wrist servo of the robotic arm moving left - remote control unit's LCD shown number (indicate robotic arm's wrist move left)
24	Press button C and push thumb joystick to right	- robotic car and hand stay still, the wrist servo of the robotic arm moving right - remote control unit's LCD shown number (indicate robotic arm's wrist move right)
25	Press button D and push thumb joystick to up	- robotic car, arm and hand stay still - remote control unit's LCD shown number (this number treated as don't care)
26	Press button D and push thumb joystick to down	- robotic car, arm and hand stay still - remote control unit's LCD shown number (this number treated as don't care)
27	Press button D and push thumb joystick to left	- robotic car and arm stay still, the robotic hand turn down - remote control unit's LCD shown number (indicate robotic hand to flip down)
28	Press button D and push thumb joystick to right	- robotic car and arm stay still, the robotic hand turn up - remote control unit's LCD shown number (indicate robotic hand to flip up)

Table 7-1-1: Test case and result of mobile robot

7.2 Fire Detection testing

7.2.1 Result and discussion

In this section, we will discuss the result of the fire detection application developed in this project using 4 videos that contain flame.

Test 1

Following diagrams show the result of using video clip 1 to test the fire detection application.

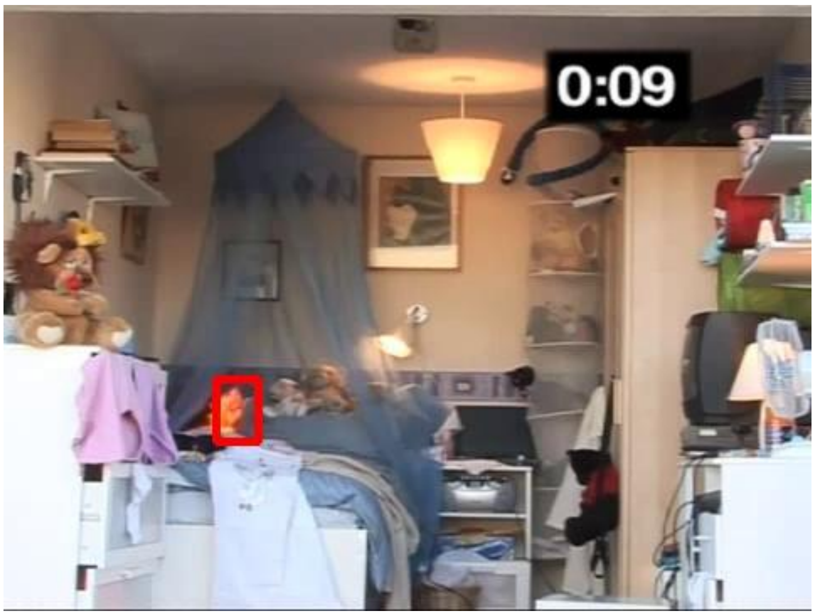


Figure 7-2-1: Test case 1 first sample

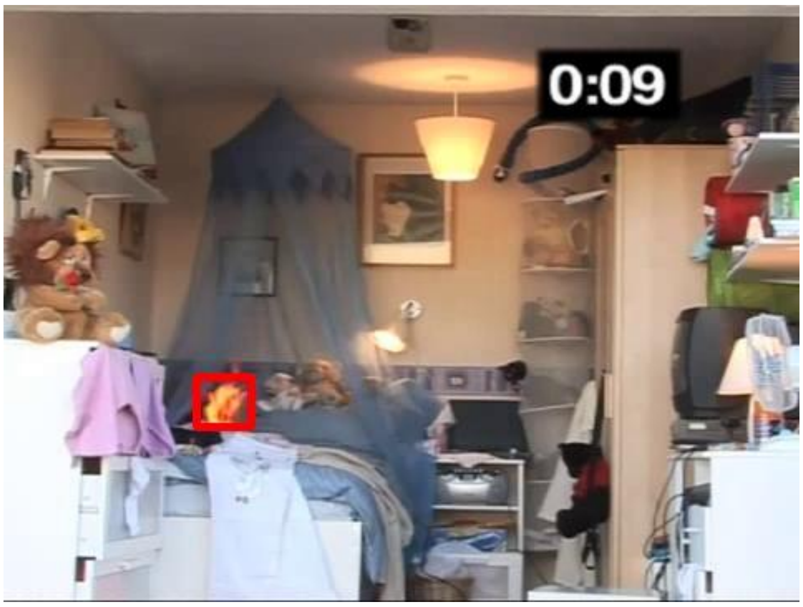


Figure 7-2-2: Test case 1 second sample

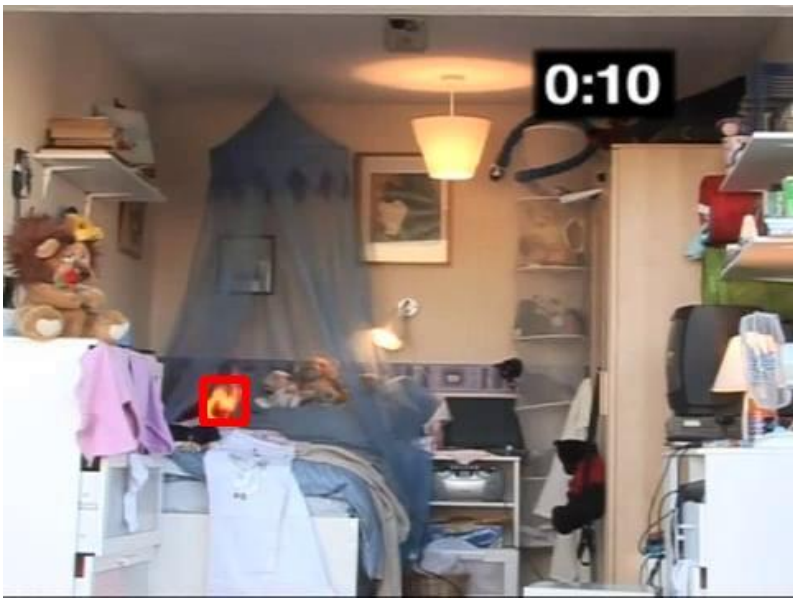


Figure 7-2-3: Test case 1 third sample

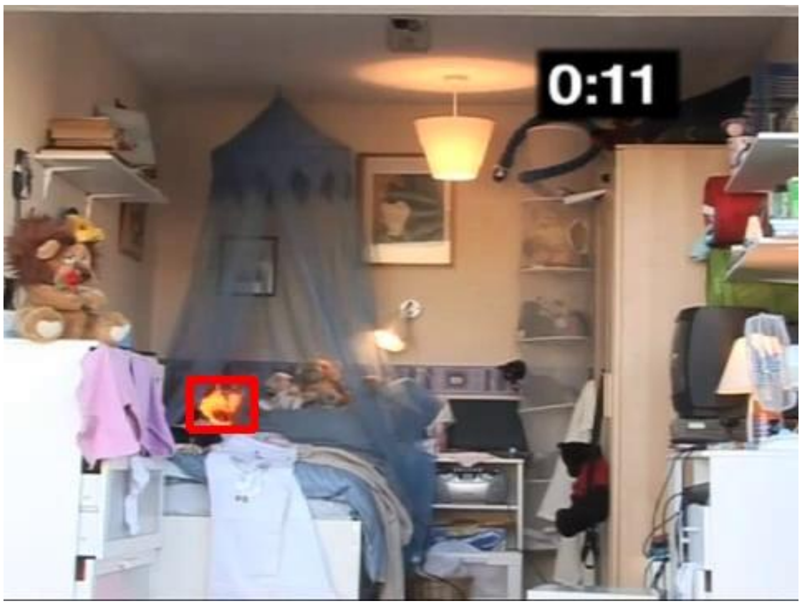


Figure 7-2-4: Test case 1 forth sample

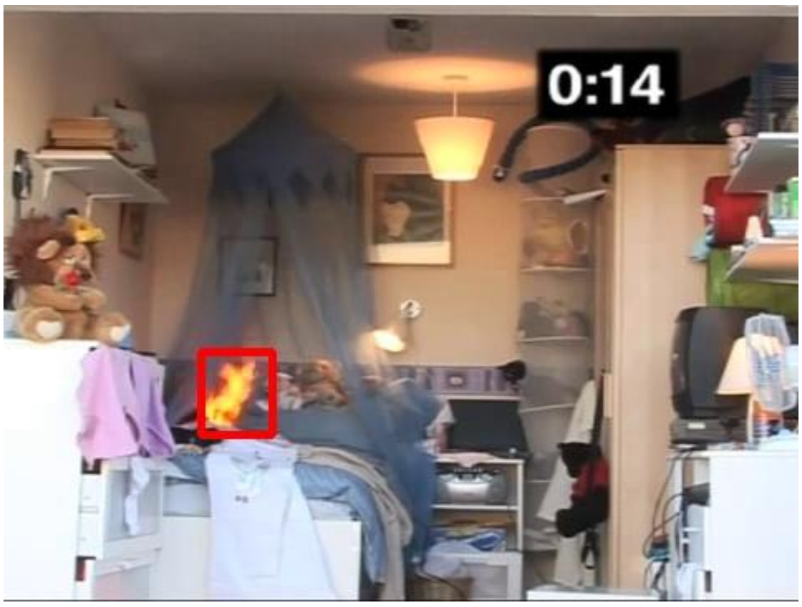


Figure 7-2-5: Test case 1 fifth sample

From all the diagrams shown, the fire can be detected clearly and whole flame been bounded by the rectangle.

Test 2

Following diagrams show the result of using video clip 2 to test the fire detection application.

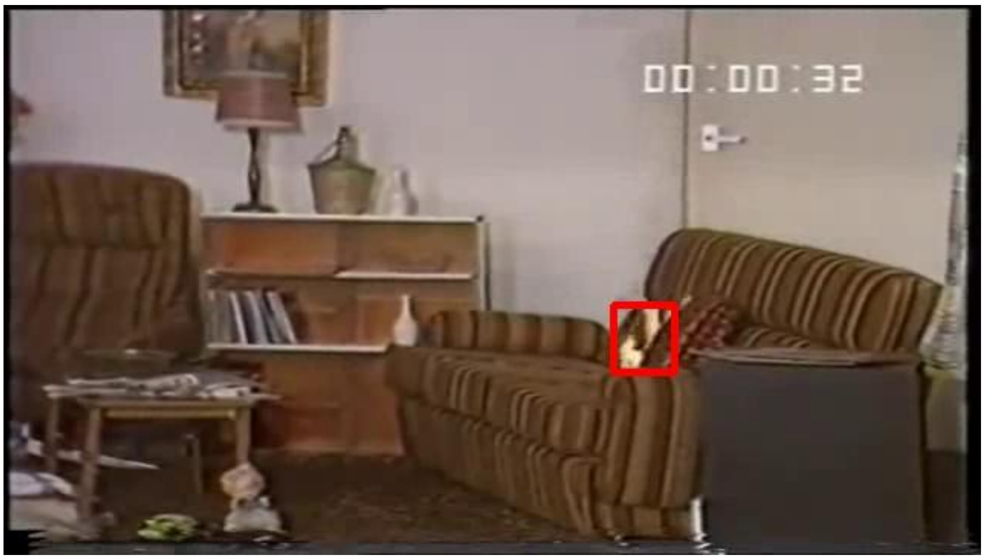


Figure 7-2-6: Test case 2 first sample

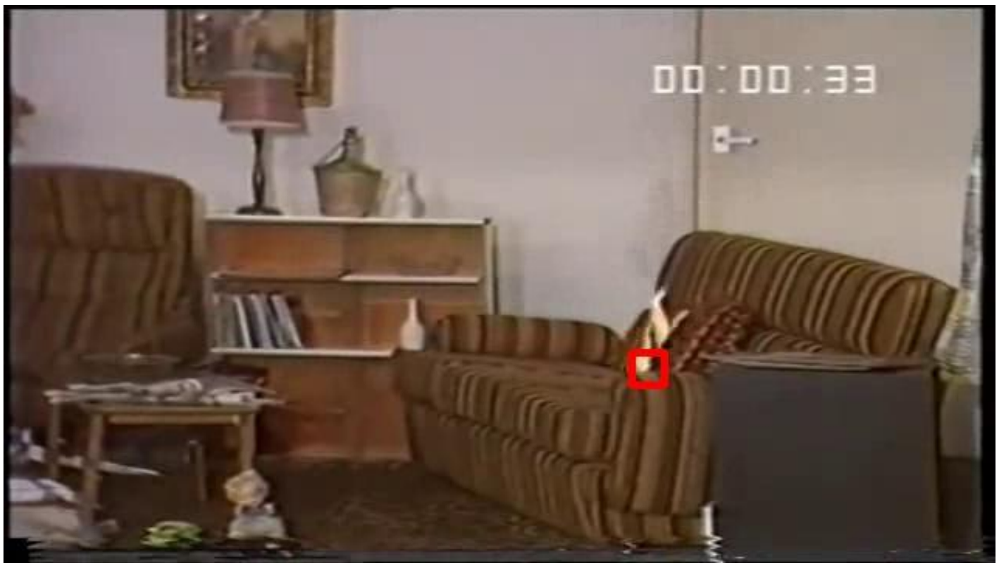


Figure 7-2-7: Test case 2 second sample

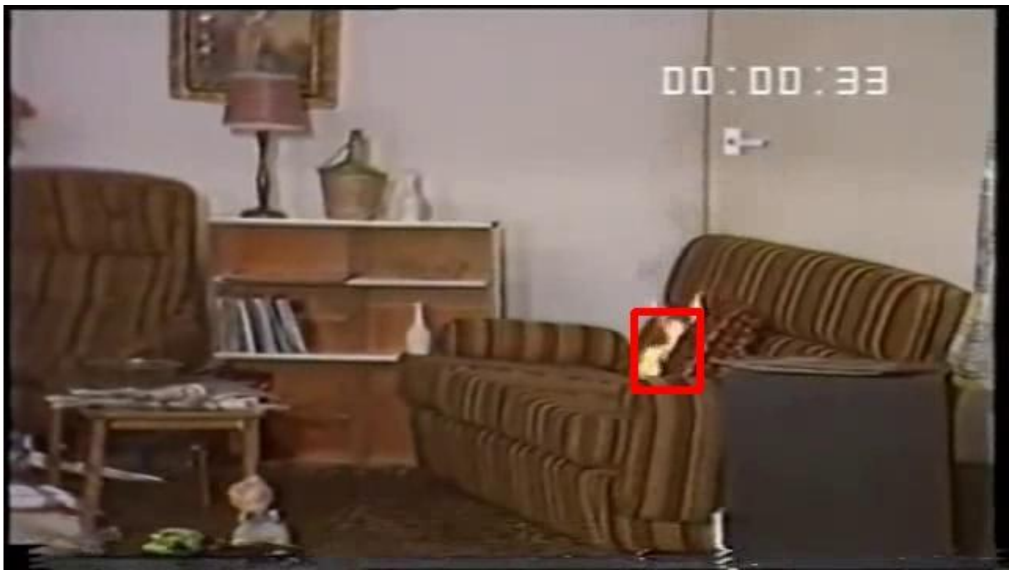


Figure 7-2-8: Test case 2 third sample

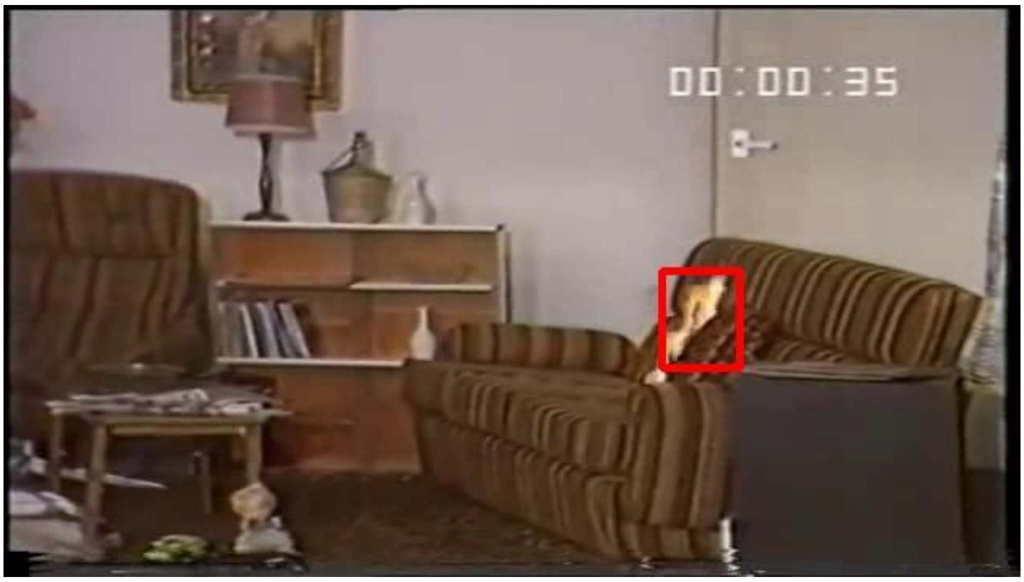


Figure 7-2-9: Test case 2 forth sample

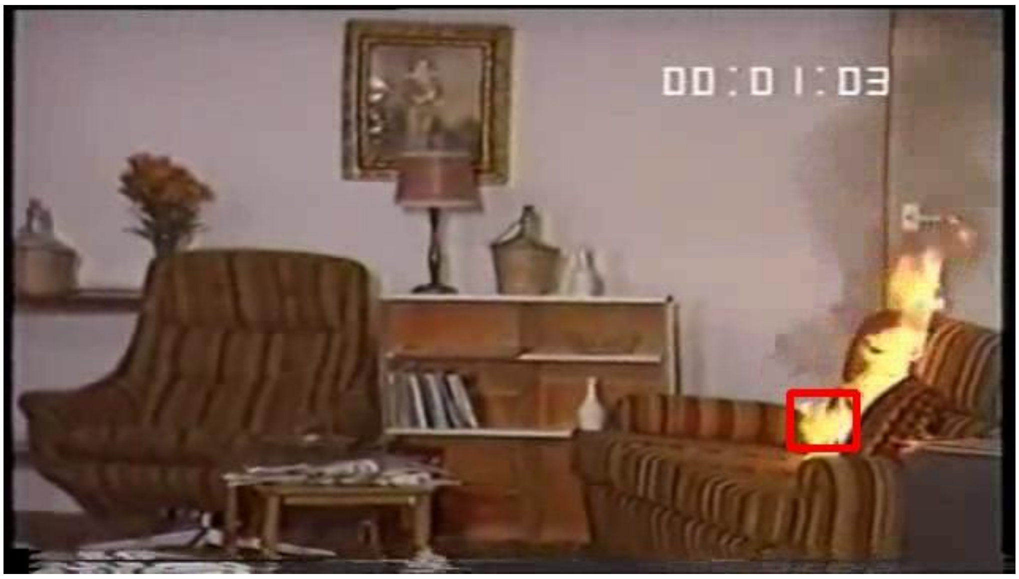


Figure 7-2-10: Test case 2 fifth sample

From all the diagrams shown, the flame can be detected clearly. However, not all the diagram can be fully bounded inside the rectangle. 2nd diagram been bounded 40% of the flame area and 5th diagram been bounded for less than 40% of the flame area.

Test 3

Following diagrams show the result of using video clip 3 to test the fire detection application.

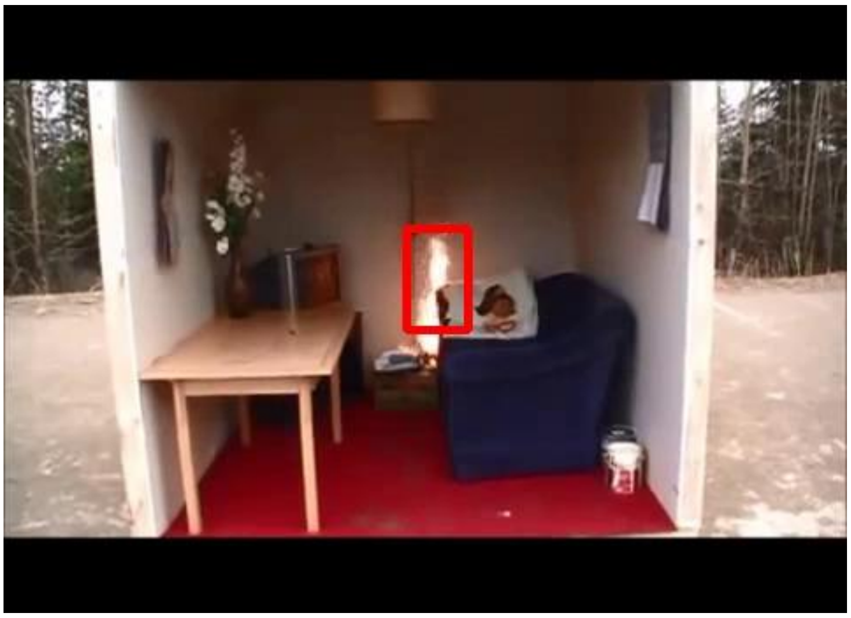


Figure 7-2-11: Test case 3 first sample

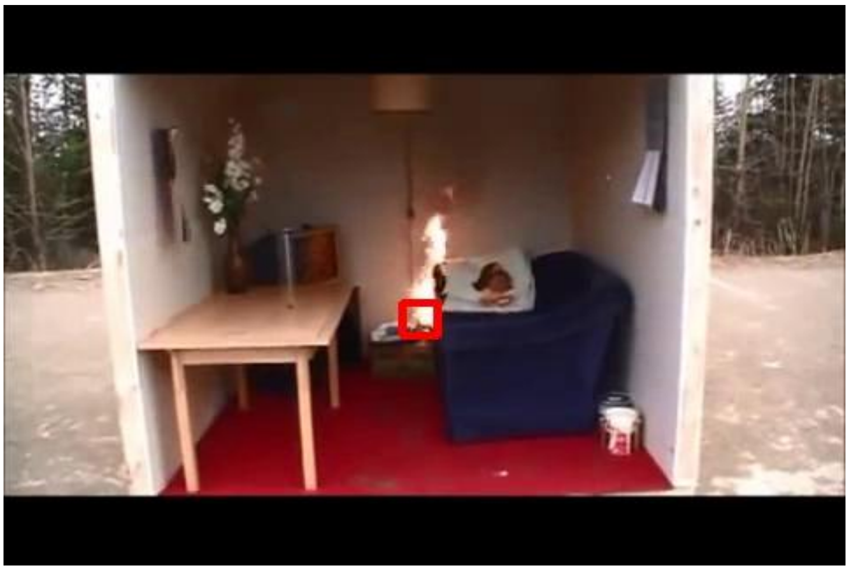


Figure 7-2-12: Test case 3 second sample

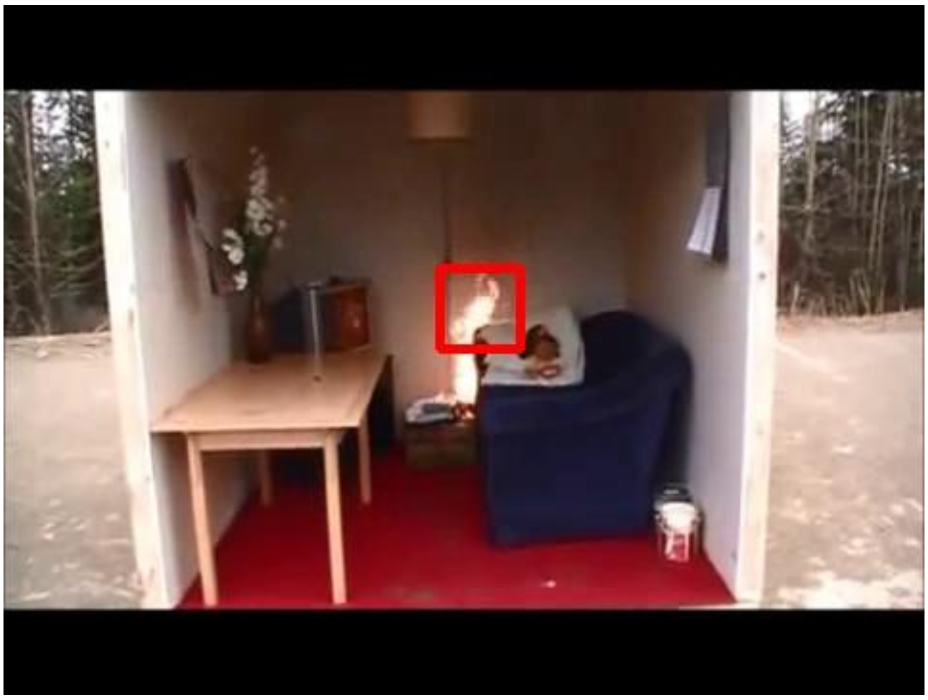


Figure 7-2-13: Test case 3 third sample

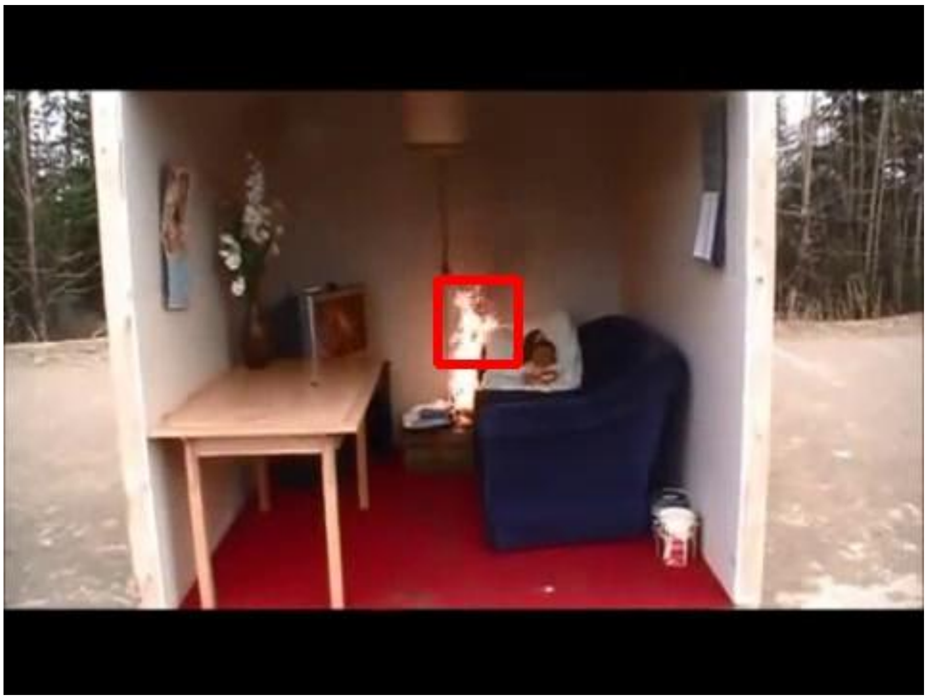


Figure 7-2-14: Test case 3 forth sample

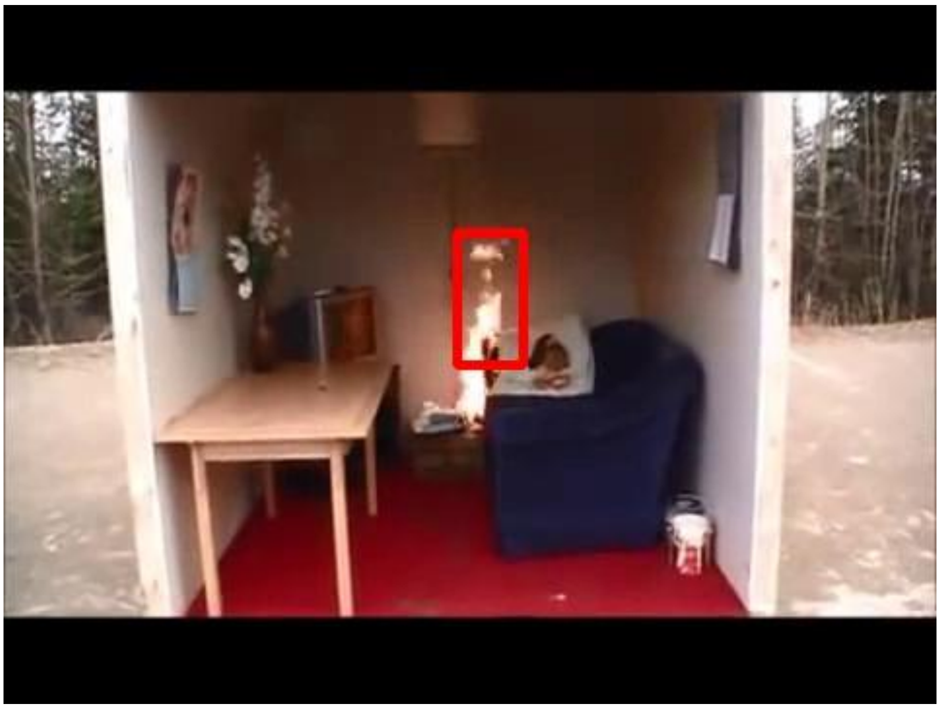


Figure 7-2-15: Test case 3 fifth sample

All the flame inside the diagrams shown above can be detected clearly. However, all of the flame area does not fully bounded inside the rectangle.

Test 4

Following diagrams show the result of using video clip 4 to test the fire detection application.

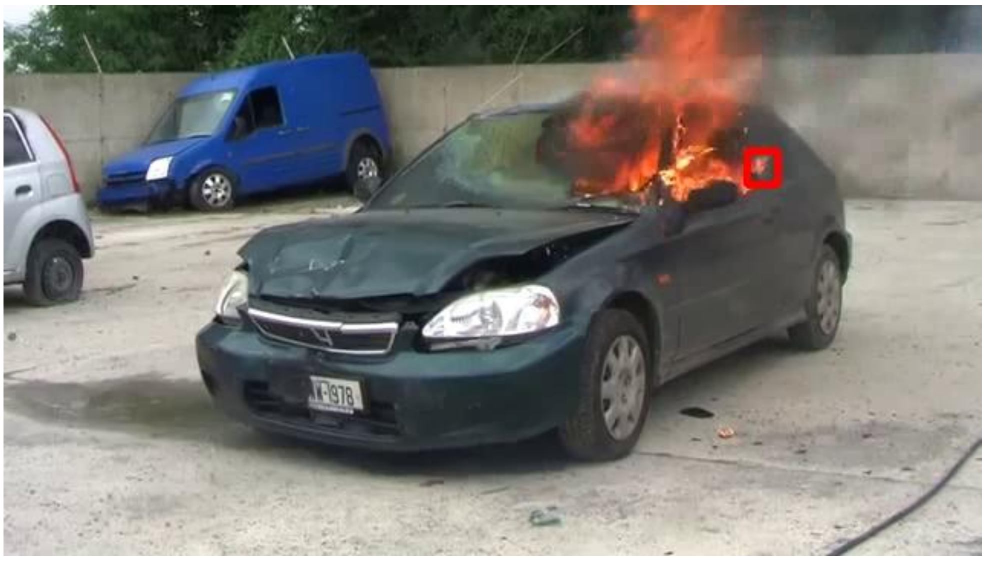


Figure 7-2-16: Test case 4 first sample

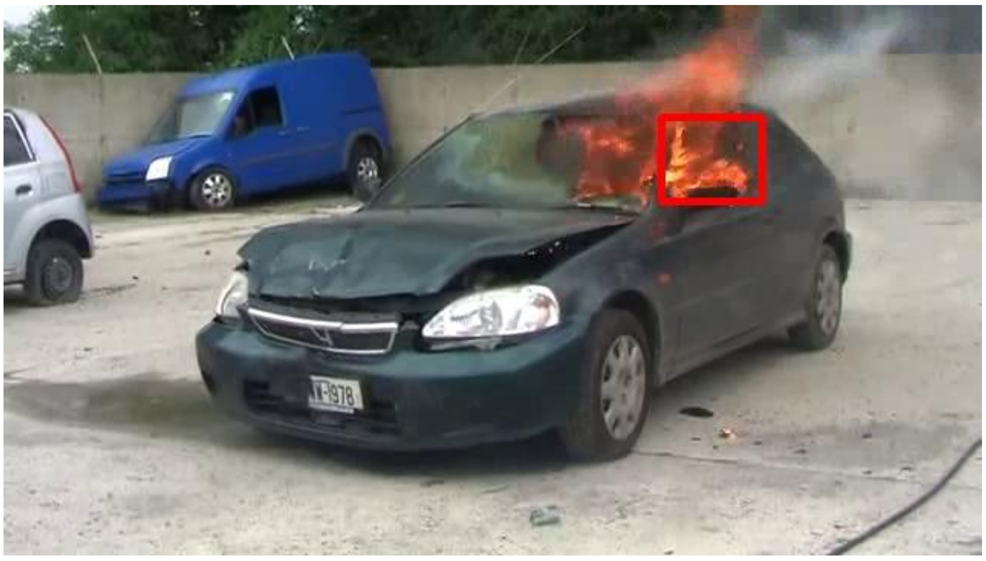


Figure 7-2-17: Test case 4 second sample

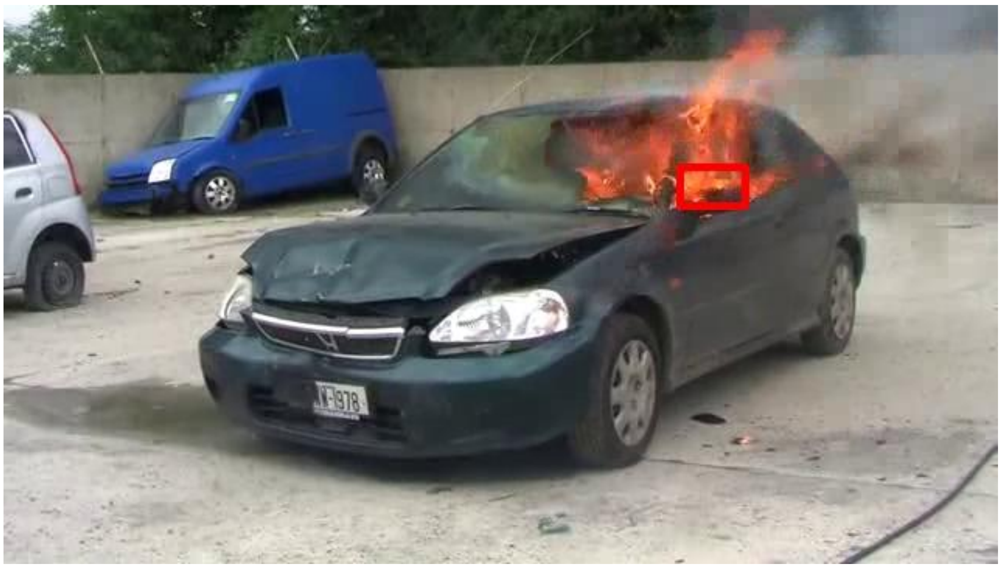


Figure 7-2-18: Test case 4 third sample

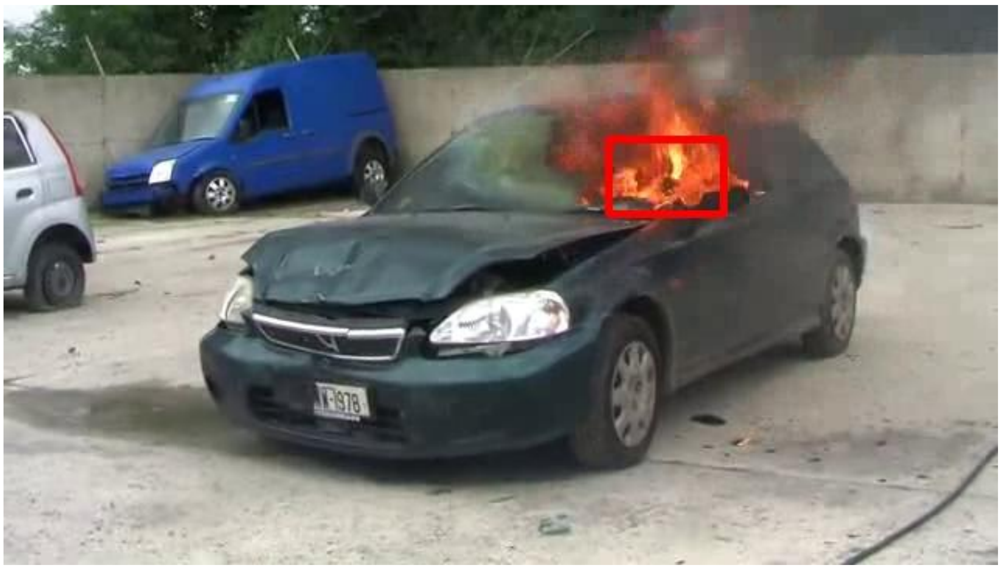


Figure 7-2-19: Test case 4 forth sample

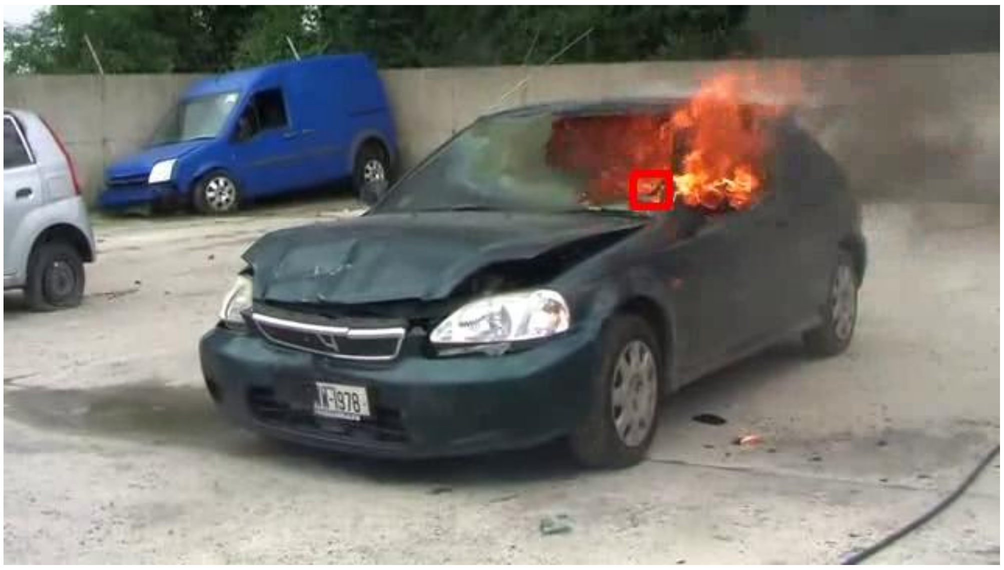


Figure 7-2-20: Test case 4 fifth sample

All the flame inside the diagrams shown above can be detected clearly. However, all of the flame area does not fully bounded inside the rectangle except that 2nd diagram and the 4th diagram the majority of the flame area.

Chapter 8 Conclusion and Discussion

Mobile robot that design in this project is able to perform dangerous task. With the robotic hand that attach to the robotic car, the direct action which is to grasps an object will be more precise and accurately. The robotic hand can act like a normal human's hand to holding, putting and grasping an object without damage the object. However, due to the robotic hand action is fully control by human through sensor glove, there are still limit in the intelligence of the robotic hand to grasp an object with its own. Furthermore, currently in market, the most popular mobile robot are all equipped with its own artificial intelligent and at the same time the development fee will getting high. With the limit of resources, the mobile robot that designed in this project would have balance between intelligence and resources.

As part of performing dangerous task, mobile robot that design in this project can used to help and assists human to avoid any dangerous issue to the human body especially for those may bring death. Furthermore, as using technologies to solve the dangerous issue is always the most wanted. Therefore, there is a demand in producing a mobile robot with high technology, efficient to perform task within limited time and effective to doing any task assign. For this project, the technology used would be moderate but will achieve both efficiency and effectiveness.

Besides that, the fire detection application that developed in this project has high efficiency and high correctness when perform to detect the fire. However, the algorithm used can only detect for yellow flame, as other research found that instead detect for flame, they do detect the smoke first. Therefore, there are still possible to improve the fire detection application and detect the flame in a faster and more accurate and efficient way.

References

1. F. Ribeiro, I. Moutinho, P. Silva, C. Fraga, N. Pereira Three Omni-directional Wheels Control On A Mobile Robot Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.99.1083&rep=rep1&type=pdf Last accessed: 27th July 2015

2. Brian Bonafilia, Nicklas Gustafsson, Per Nyman, Sebastian Nilsson Self-balancing Two-wheeled Robot Available: http://sebastiannilsson.com/wp-content/uploads/2013/05/Self-balancing-two-wheeled-robot-report.pdf Last accessed: 27th July 2015

3. Viboon Sangveraphunsiri, Mongkol Thianwiboon Traction Control of a Six-wheel Robot using Wheel Slip Dynamics Available: http://www.regional-robotics.org/lab_info/Paper/Journal/Traction%20Control%20of%20a%20Six-wheel%20Robot%20using%20Wheel%20Slip%20Dynamics.pdf Last accessed: 27th July 2015

4. Lael U. Odhner, Leif P. Jentoft, Mark R. Claffee, Nicholas Corson, Yaroslav Tenzer, Raymond R.Ma1, Martin Buehler, Robert Kohout,Robert D. Howe, Aaron M. Dollar (2013) A Compliant, Underactuated Hand for Robust Manipulation Available: http://arxiv.org/ftp/arxiv/papers/1301/1301.4394.pdf Last accessed: 27th July 2015

5. Raphael Deimel, Oliver Brock (2013) A Compliant Hand Based on a Novel Pneumatic Actuator Available: https://www.robotics.tu-berlin.de/fileadmin/fg170/Publikationen_pdf/2013-icra13_Deimel_Brock.pdf Last accessed: 27th July 2015

6. Anthony L. Crawford, Jeffrey Molitor, Alba Perez-Gracia, Steve C. Chiu Design of a Robotic Hand and Simple EMG Input Controller with a Biologically-Inspired Parallel Actuation System for Prosthetic Applications Available:

http://digital.csic.es/bitstream/10261/97426/1/Design%20of%20a%20robotic%20hand.pdf Last accessed: 27th July 2015

7. Ankit Gupta, Mridul Gupta, Neelakshi Bajpai, Pooja Gupta, Prashant Singh (2013) Efficient Design and Implementation of 4-Degree of Freedom Robotic Arm Available: http://www.ijeat.org/attachments/File/v2i5/E1720062513.pdf Last accessed: 27th July 2015

8. Riyaz Mansuri, Sandesh Vakale, Ashish Shinde, Tanveer Patel (2013) Hand Gesture Control Robot Vehicle Available: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=5EB9FCE0F8A1B119E8DF5CDB63919310?doi=10.1.1.416.8918&rep=rep1&type=pdf Last accessed: 27th July 2015

9. Shamsheer Verma (2013) Hand Gestures Remote Controlled Robotic Arm Available: http://www.ripublication.com/aeee/80_pp%20%20%20%20601-606.pdf Last accessed: 27th July 2015

10. Nazrul H. Adnan, Khairunizam Wan, Shahriman A.B., M. Hazwan Ali, M. Nasir Ayob, Azri A. Aziz (2012) Development of Low Cost "GloveMAP" Based on Fingertip Bending Tracking Techniques for Virtual Interaction Available: http://www.ijens.org/vol_12_i_04/129704-5858-ijmme-ijens.pdf Last accessed: 27th July 2015

11. Vipin V (2012) Image Processing Based Forest Fire Detection Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.414.3668&rep=rep1&type=pdf Last accessed: 27th July 2015

12. Jareerat Seebamrungsat, Suphachai Praising, Panomkhawn Riyamongkol (2014) Fire Detection in the Buildings Using Image Processing Available: http://www.ispc.ict.mahidol.ac.th/documents/ICT-ISPC2014%20Proceeding/data/ISPC2014_4.pdf Last accessed: 27th July 2015