AGHILAN NARAYANAN B. ENG. (HONS) ELECTRICAL & ELECTRONICS ENGINEERING MAY 2014

EMBEDDED SYSTEM OBJECT TRACKING USING CMU CAMERA

AGHILAN A/L NARAYANAN

13939

MR. PATRICK SEBASTIAN

ELECTRICAL & ELECTRONICS ENGINEERING

UNIVERSITI TEKNOLOGI PETRONAS

MAY 2014

Embedded System Object Tracking using CMU Camera

By

AGHILAN A/L NARAYANAN

13939

FINAL PROJECT REPORT

Submitted to the Electrical & Electronics Engineering Programme

in partial fulfilment of the requirements

for the Degree

Bachelor of Engineering (Hons)

(Electrical and Electronics Engineering)

Universiti Teknologi PETRONAS

Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Embedded System Object Tracking using CMU Camera

by

Aghilan a/l Narayanan

13939

A project dissertation submitted to the Electrical and Electronics Engineering Programme

Universiti Teknologi PETRONAS

in partial fulfilment of the requirement for the

Bachelor of Engineering (Hons)

(Electronics & Electrical Engineering)

Approved by,

_____

Mr. Patrick Sebastian

Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

May 2014

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

_____

AGHILAN NARAYANAN

ABSTRACT

Computer vision has gone long way from 1980s till now. Before, computer vision generally so robust and expensive. However, due to advancement in embedded processing, computer vision can be implemented in cheaper computer system. Raspberry Pi and CMUcam4 provides fantastic platform for interfacing and also provides a simple robotic object motion tracking at school-level or university-level. These simple computer vision systems can benefits in many ways such as being applied to police task force, or bomb squad purpose, or firefighting purpose. To achieve the interfacing, serial communication between the RPi and CMUcam need to be achieved. Then we'll develop the color tracking algorithm to track a color and ask the camera to follow that colored object movement. From the troubleshooting and findings, interfacing RPi and CMUcam requires a lot of tweaking as both consists of two different systems. In the end, only servo platform does not successfully work.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratefulness to the Almighty God for showing me the path and meeting important person that allow me to do the project properly. I would like to express high gratitude to any individuals who always gave their precious time to attend to me.

Special thanks to my supervisor, Mr. Patrick Sebastian for his guidance and assistance during my project duration. His encouragement and advices is what pushed me to try harder during the project work. Thanks to the Final Year Project Committees for their information and guidance given for the project.

I would like to express my thanks to all the lab technician, lecturers and friends who had provided many valuable advices throughout the project.

Last but not least, I would like to thank my parent for their understanding and patience with my project.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

RPi = Raspberry Pi

CMUcam = Carnegie-Mellon University camera

UART = Universal Asynchronous Receiver Transmitter

USB = Universal Serial Bus

RX = Receiver

TX = Transmitter

PWM = Pulse Width Modulation

mx = average of all tracked x-coordinates pixels.

my = average of all tracked y-coordinates pixes.

x1 = first x-coordinate of the tracked pixels

x2= second x-coordinate of the tracked pixels

y1= first y-coordinate of the tracked pixels

y2= second y-coordinate of the tracked pixels

CHAPTER 1

INTRODUCTION

1.1 Background of Study

In the current modern technology world, the computer system getting more and more advanced, with implementation of image processing, video processing, and simplification of data size. The concepts of every computer systems are similar, that is a complete, working computer. It includes the computer along the software and peripherals devices that makes the computer function properly. [1]

As a result of the advancement in computer system world, the robotics area is also developed in rapid progress. The robotic applications are now being included in people's daily life and not only catered for industrial life. Most of the robotic applications are being used for military, medical and educational purposes. In the old time, robots are generally large, and taking a lot of wiring and troubleshooting to make it function properly and such large robots are usually used to handle heavy work. However, as time progresses, the robot applications start to become more mobile thanks to the embedded processing and system-on-chip (SoC) as well as the introduction of small size electronic components. Moreover, the type of robots that usually made is autonomous type, which means it have the ability to make decision with help of the pre-programmed functions in their hardware. The autonomous robot usually has sensors, in order to connect to outside world. These sensors are required in order to bring senses to the robot that is vision, and audio. Current image and video processing is typically done in the complete computer system, and the said computer system is not exactly cheap and mobile.

This final year project aimed to use embedded system and CMUcam to track an object movement. Raspberry Pi (RPi) and CMUcam4 will be used as embedded

system and image processing unit respectively. RPi is relatively new in the technology world, and yet it becomes so popular that every time a batch of several thousand units is produced, it will be sold out within a very short time frame. RPi is basically a credit-card sized complete computer system, which can run an Operating System (OS) on the board itself. [2] Provided that it can function as a normal PC, it is relatively cheap compared to the PC. The retail price of RPi is 5 times less than the price of normal PC. Besides, RPi has outlet to interface with other devices such as microcontrollers and the motor servos through serial communication. CMUcam4 is a camera sensor which has its own processor on the board. It is so powerful that it can do basic image processing without any external circuitry by using several commands that can be applied in programming. However, the current RPi only provide 3.3 V DC power to the outlet, so several advanced outlet cannot really connect with RPi.

Since a lot of programming, image processing algorithm and electronics knowledge are required, it provides a good learning tool for robotics area. Further study will eventually develop a more complete image processing on cheap computer system like the CMUcam and RPi combo for bomb defusing skill, firefighting, criminal prevention program, and various other purposes.

1.2 Problem Statement

In this final year project, several aspects need to be addressed before it can be successfully demonstrated to public. First of all, the method of colour tracking algorithm should be decided so it can be processed as close as possible in real time. Since the image processing algorithm and the servo control have delay times, we need to program both the CMUcam4 and servo and RPi to work as close as possible in the real time tracking. This probably can be achieved by using a method that will not take a long image processing time.

The second aspect is to use Raspberry Pi and CMUcam together through interfacing. Since RPi is relatively new in the market, interfacing both of them is rarely done or never done before and therefore provide an open obstacle, which is to find a solution and way to interface both of them. Programming using C or Python language and GPIO know-how will probably best way to overcome the interfacing

problem. Provided that both CMUcam4 and RPi are two different systems, it proves to be real challenges in interfacing them.

The third aspect is to connect to a good platform so that it able to track object movement flawlessly. A good platform means that it needs a good and easy to use servo or motor in order to track an object.

Last but not least, overall cost of production should be low compared to available embedded system object tracker in the market as this is the reason of the existence of this final year project.

1.3 Objective

The objective of this project is to build an embedded system object tracking using Raspberry Pi and CMUcam4. There are two main functions, to track an object and to move according to the object movement.

1.4 Scope of Study

The project mainly focuses on object tracking using Raspberry Pi and CMUcam4. This includes interfacing the RPi and CMUcam4 by means of serial communication. Since both RPi and CMUcam4 are of different systems, more study on the schematic and board layout is needed before making any connection.

The next step is to do programming on the RPi and CMUcam4 to allow both to work together in image processing. We need to learn C/Python programming and GPIO know how as well as the CMUcam command list in order to do complete programming. The next step will be in interfacing the now complete RPi+CMUcam combination to be interfaced with servo platform.

The last step is to develop an algorithm to track the movement of an object. The movement of the object will be done by means of color tracking and from there on will be developed into more elaborate tracking which hopefully will be including the predictive movement of the object.

CHAPTER 2

LITERATURE REVIEW

2.1 Raspberry Pi as Embedded System

In the year 2011, RPi took the electronic world by storm. It is showcased as the credit-card sized computer system, with ARM-11, an Ethernet port, USB port, GPIO port, HDMI port, a 512MB memory and a SD card slot. All this goodness is in one board with price of £25 (~RM115). According to the Eben Upton in his latest conversation as documented in "Computing Conversations Magazine", he's been working on introducing a cheap computer to reinstate the interest in computer science and to provide a platform for people to work on their creativity since 2005. [3] Since a more complete computer system is more expensive, most of the people specialised in the computer science always take a risk on trying to tinkering with it. Now that RPi has been introduced, it has been extensively used in developing many applications and in many ways, such as being a GUI interface on an old microwave. [3] Nowadays, all the mobile robots are being based on the Raspberry Pi since it provides a more complete coverage and programming tinkering. In another interview, Crispin Andrews said that they wanted a computer that people can do things with. [4] This was of course backed up by the fact that one million units of RPi have been shipped worldwide.

It is possible to connect the RPi to outside world thanks to implementation of UART in the system. It is the key components which enables a serial between devices. The following shows the UART of GPIO pin on the RPi.

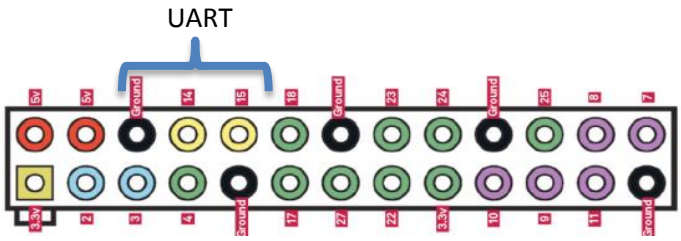

Figure 1: Taken and Modified from Linux and User Developer Issue 132 (2013)

2.2 Image Processing and Object Tracking

Simple computer vision algorithm proves to be extremely important in many areas, which include military, sensing area, educational, robotics, medical and various other areas. [5] [6] [7] [8] Since traditional vision system requires a camera, frame grabber and a high speed processor, it proves to be real challenging in implementing computer vision using embedded systems. However it becomes more possible as there is availability of CMOS color camera modules like CMUcam4 and low cost microcontrollers or computer system like RPi. Image process usually takes a lot of process which in turn process a lot of computing power. However, thanks to image buffering ability provided by the embedded system chip, the power consumption and the image compression size are minimized. There are systems that based on the image processing and sensing available in the market, but they are usually too expensive to be done commercially and in bulk. For example, LIDAR and stereo vision system which can stands at almost $40,000. [9]

Object tracking using low-cost camera module is now starts to be growing in demands. Object tracking using computer vision is already done for a long time. However, price seems to be the limiting factor here. For example, Newton Lab's Cognachrome system which is a computer vision sensors, costs around 2500 to 3500 dollars per unit. [10] Ability to track object with embedded system will benefit many areas, such as the firefighting area, bomb defusing squad, or rescuing squad and provides a huge cost effective alternative compared to current market products.

There are several methods used in image processing. For example, a research uses Unscented Kalman Filter and Gaussian filter to process image. [11] [12] Most recently, YCbCr method is the most commonly used in color space. This technique detects the object by obtaining the threshold of a certain color. By getting the

threshold of the color using CMUcam4, we then can implement the code into RPi to make it to track the object motion.

CHAPTER 3

METHODOLOGY

3.1 Research Methodology

The flowchart below shows the overview of the research methodology in this paper. The explanation is located in the next sub-chapter, project activities.

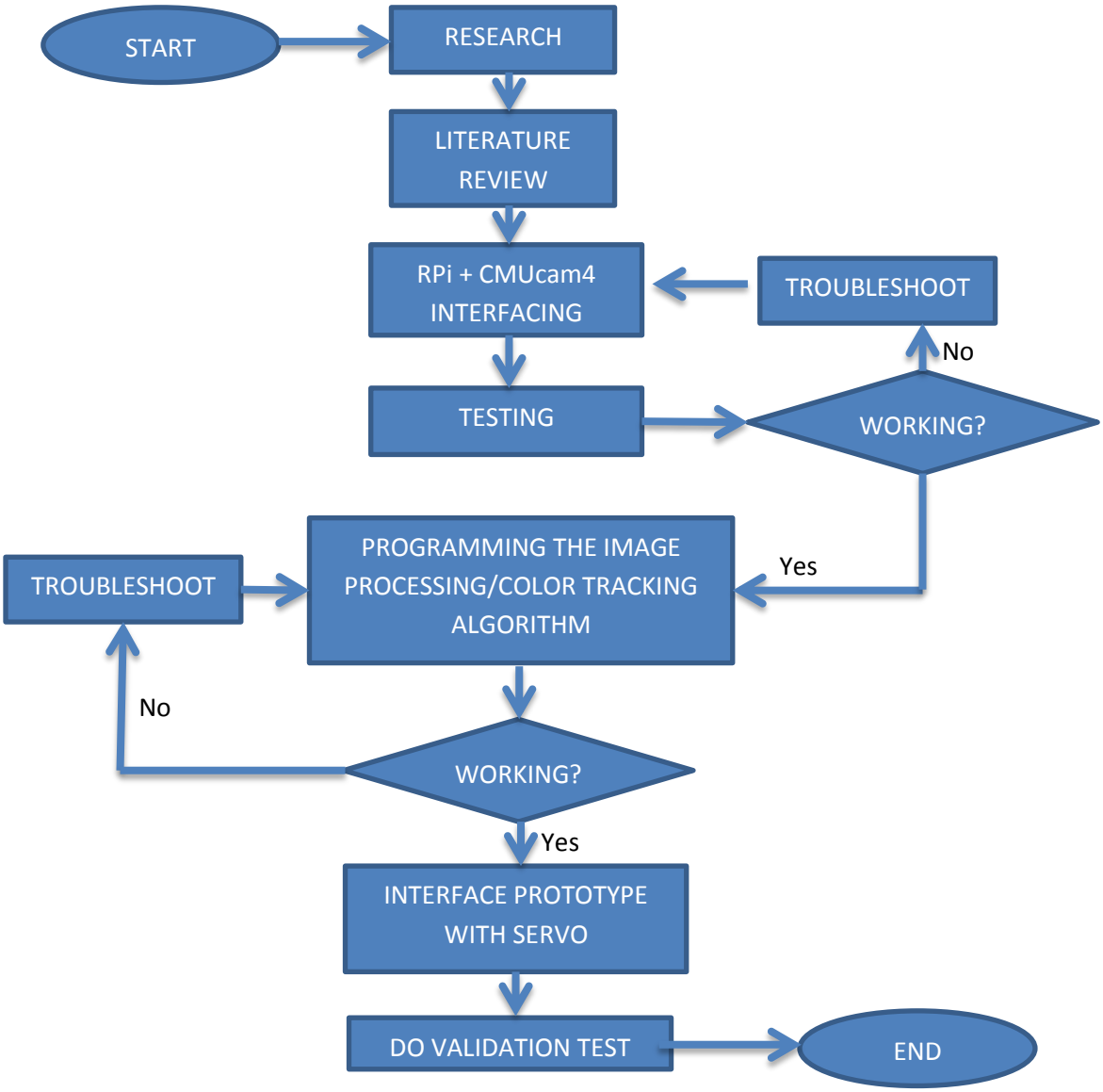

Figure 2: Methodology Flowchart

3.2 Project Methodology

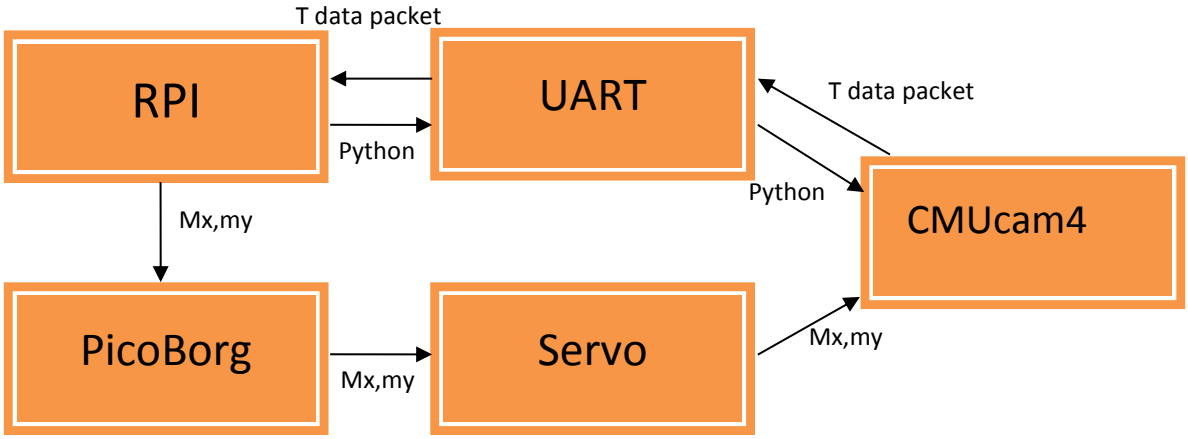

Figure 3: Data Movement Methodology

Based on the figure above, RPi will send instructions to CMUcam4 through UART in Python Language. The CMUcam4 then process the program and carry out the intended tasks and then produces T data packet in form of [T mx my x1 y1 x2 y2 pixels confidence] to be displayed in the RPi's terminal. The RPi is then extract "mx" and "my" value to be fed into the servo for positioning. The T data packet in form of [T mx my x1 y1 x2 y2 pixels confidence] is shown in the Figure 4 in practice. Figure 5 shows the servo positioning. The planned positioning is as follow.

Case 1: If mx is equal to 30º, the servo needs to move to right until mx is centered. The same goes to my coordinates.

Case 2: If mx is equal to 110º, the servo needs to move to left until mx is centered.

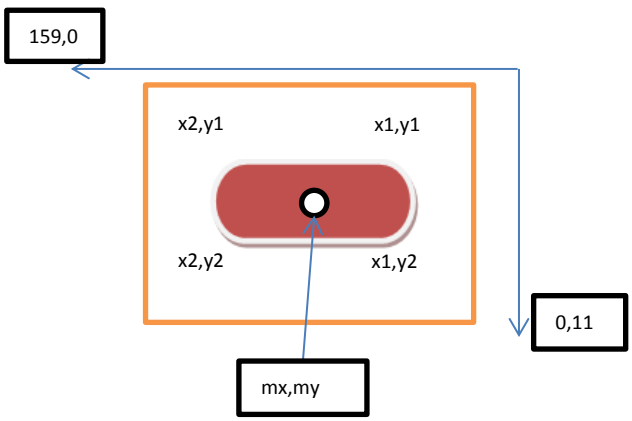

Figure 4: Camera Layout

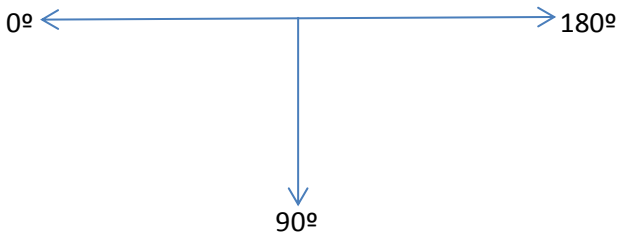

Figure 5: Servo Positioning

3.3 Project Activities

The Research part is done on collecting more information on low-cost camera module and how to interface them with the low-cost microprocessor. In the research, all the sources of information are properly viewed and documented.

Literature Review part focuses on preparing the background of study, outlining the problem statements, and then narrowing down the topic of interest on the project.

Then interfacing RPi and CMUcam4 will be done to perform specific task. The system must be error-free and must be able to communicate with each other. Troubleshoot any problem that appeared in the process by gathering more information and by asking expert. Raspberry Pi will act as the microprocessor that communicate and tell the CMUcam4 the order. It also will act to get the image from CMUcam4. CMUcam4 have various built in functions and needed to be tested under various conditions, like lighting, or the surroundings.

Once interfacing done, programming is required to implement the image processing and color tracking algorithm which will be used as basis of object motion tracking. More troubleshooting will be expected in order to make the system functioning properly. Programming in C will be extensively used throughout the project since it is more hardware friendly.

The next step will be to interface the RPi and CMUcam4 with servo system to enable more extensive camera coverage while tracking object motion. The servo consists of pan/tilt header that will move according to the object movement. The last step is to do validation test in order to call the whole project as a success.

3.4 Key Milestones

The following flowchart shows the key milestones that are achieved during the weeks until the submission of Progress Report.

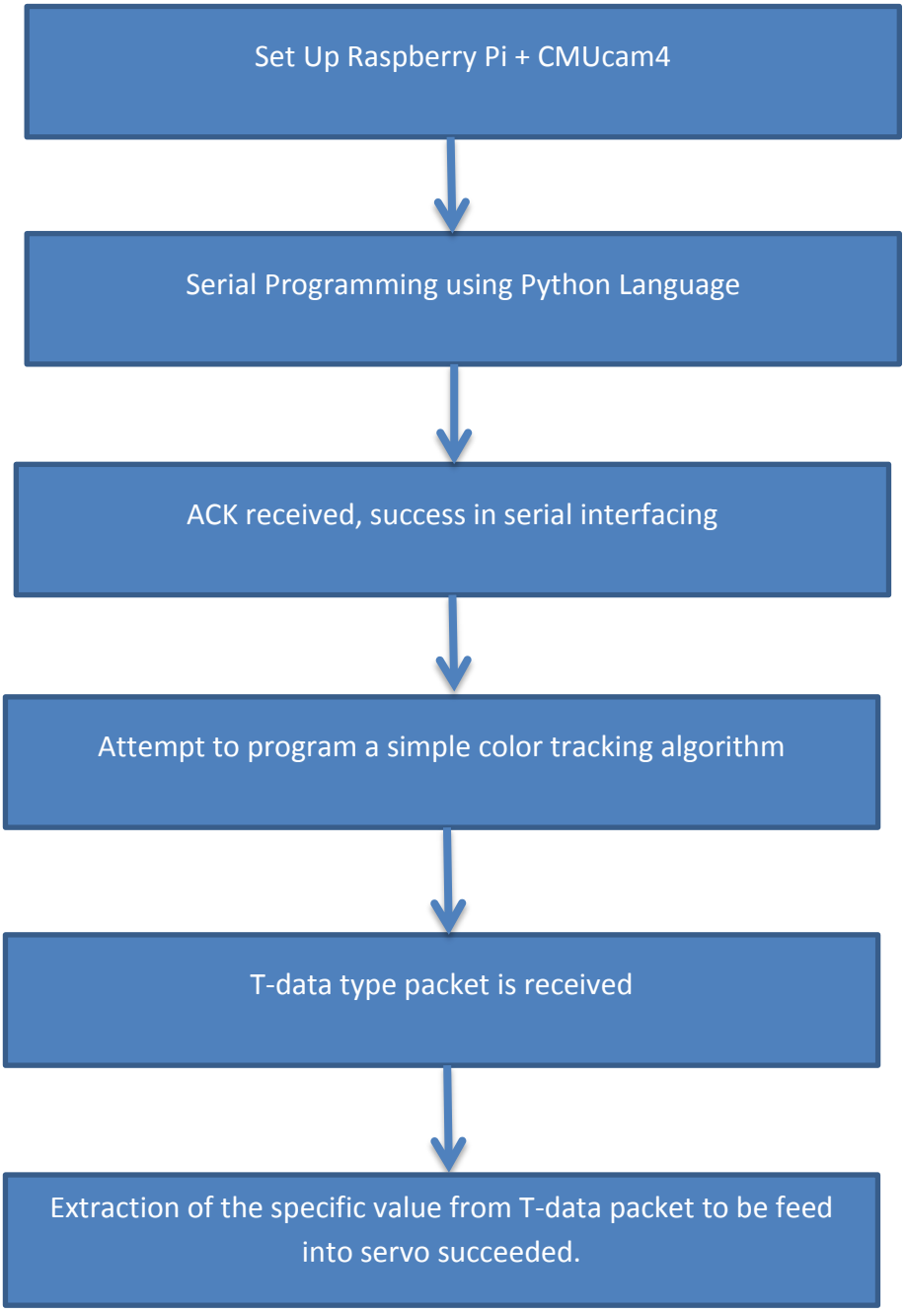

Figure 6: Key Milestones Flowchart

3.5 Gantt Chart

Table 1: Gantt Chart

Project Schedule	Weeks No.														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
System Interfacing															
System Programming															
Progress Report															
PRE-SEDEX															
Draft Dissertation															
Dissertation (softbound)															
Viva															
Dissertation (hardbound)															

CHAPTER 4

RESULT AND DISCUSSION

4.1 Setup of the FYP

4.1.1 Initial RPi Setup

In the project, the following is needed for setting up the FYP. In the figures below, the UART to USB connector is connected to USB port of Raspberry Pi and UART connector of CMUcam4.

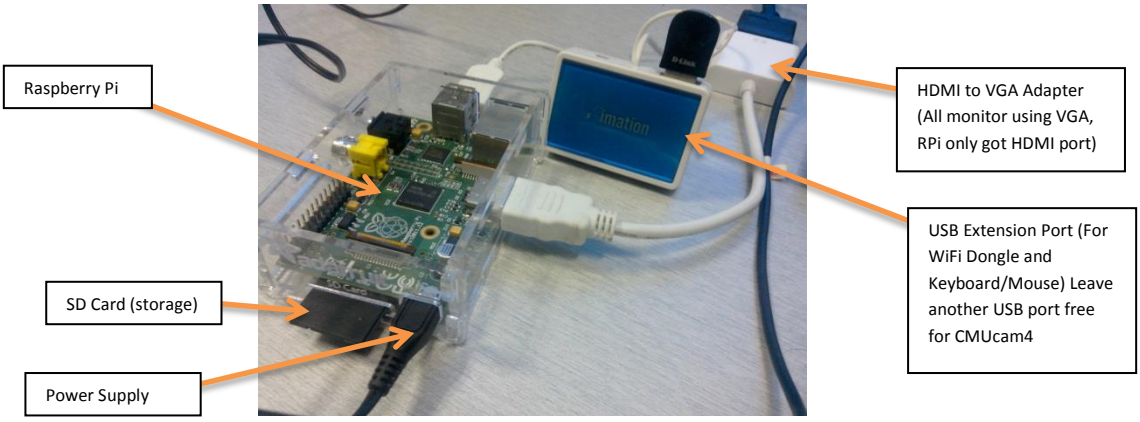

Figure 7: Raspberry Pi Setup

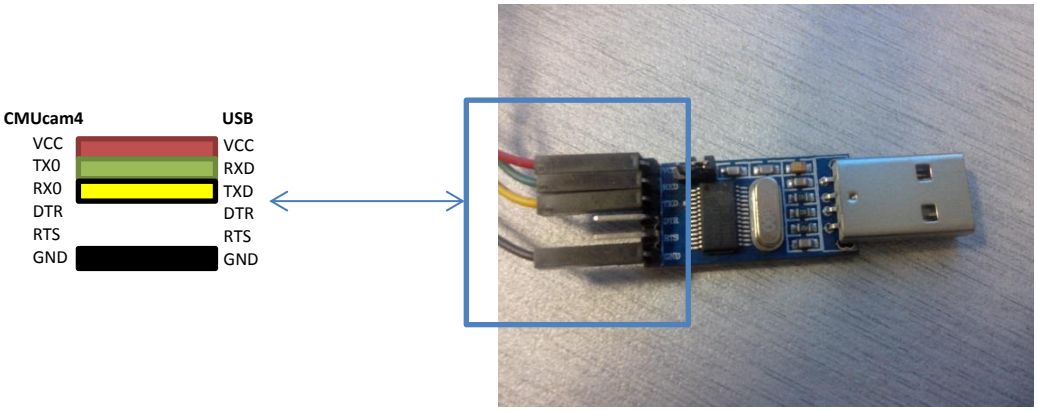

Figure 8: UART to USB Adapter and the connection

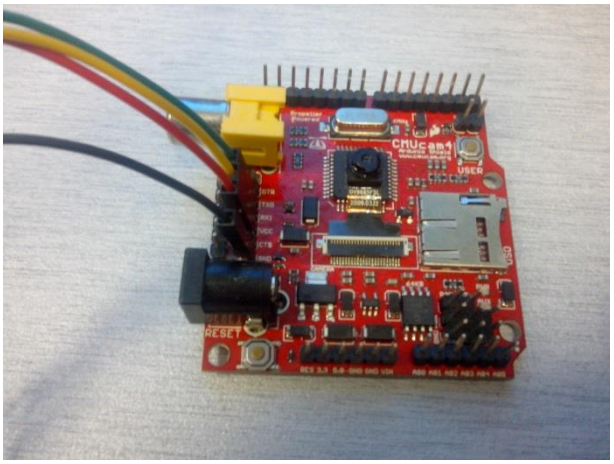

Figure 9: CMUcam4

The following screenshot shows the successful connection of Raspberry Pi and CMUcam4GUI.

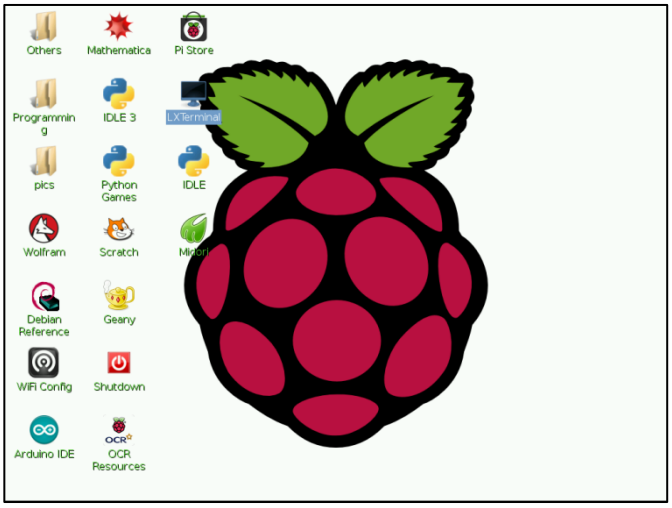

Figure 10: Raspberry Pi Screen

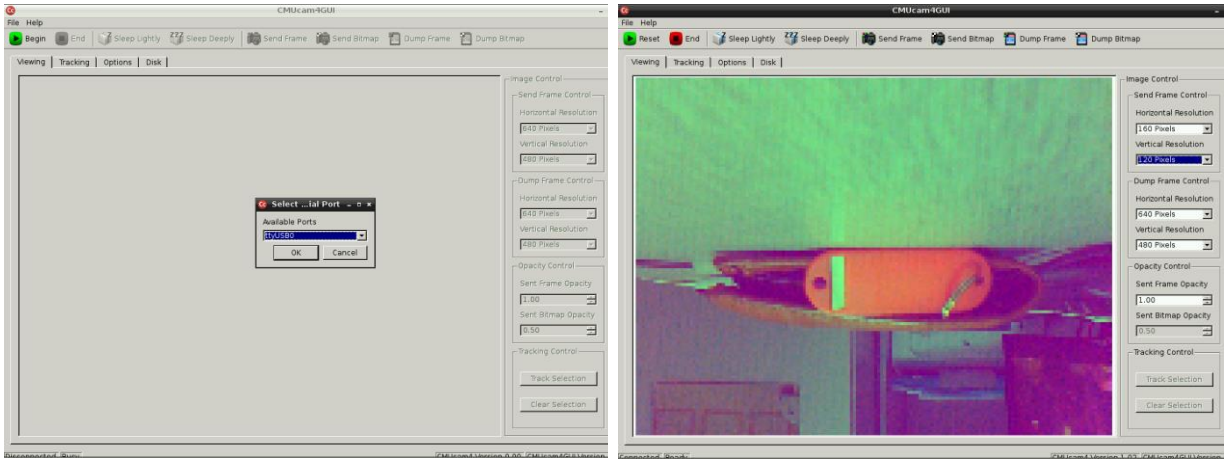

Figure 11: CMUcam4GUI start screen and after capturing an image

Several setting of RPi is tweaked for purpose of displaying it using an external monitor. This involves setting the display brightness, the hotplugging of monitor, the setting of RPi network for access of file using my laptop as well.

4.1.2 Setup of Serial Communication

There is several way tested in order to serially connect to CMUcam4. First was to connect the jumper into the CMUcam4 UART to Pi UART. The following setting is tweaked.

Backup the /boot/cmdline.txt file before editing(in case things go wrong)

- sudo cp/boot/cmdline.txt /boot/cmdline_backup.txt

Edit the file

- sudo nano /boot/cmdline.txt

Delete parameters involving the serial port "ttyAMA0"

- console=ttyAMA0, 115200 kgdboc=tyAMA0, 115200

Save and Close. And edit file:

- sudo nano /etc/inittab

Search for serial port usage by typing /ttyAMA0/ and comment it out by putting "#" at the beginning of the text and reboot

The next step is to set up a terminal console, in this case, using SimpleIDE for the communication. The following figure shows the ACK received.

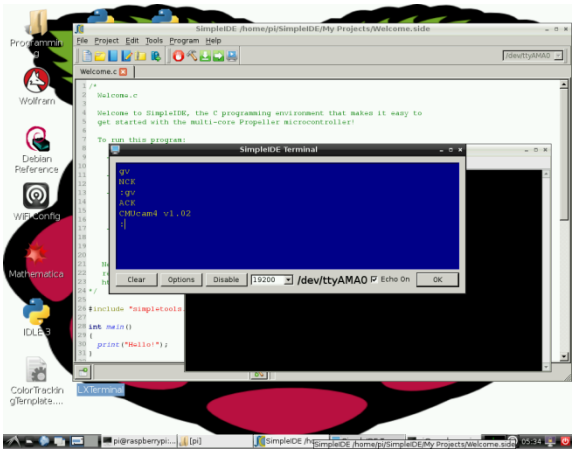

Figure 12: ACK Received

However it stops at this. Beyond the “GV” command, we are unable to give any other instructions and sometimes the camera freezes.

In the end the UART-USB is used whereby the CMUcam4 is connected using UART and USB is connected to the Raspberry Pi as shown in Figure 8 and 9. This method is used indefinitely since all the instructions and commands available in the CMUcam4 Command List V02 can be passed to CMUcam4. CMUcam4GUI is successfully set up as shown in the Figure 11 which proves that the serial communication is successful.

4.1.3 Setup of PWM for usage of LED and Servo

In the Pi, it is stated that it only contains Hardware PWM which is pin 17 of its GPIO connector. Several ways is tested to connect the Hardware PWM to the LED first. Blinking of LED is successful since the pin output required voltage and using C language and Python language are both successful. However, connecting servo to it is not successful therefore it is requiring a logic level shifter from 3.3V to 5V. PicoBorg serves the purpose as the logic level shifter in this case, allowing the servo to move 0º, 90º and 180º within a given time. However setting the servo using PWM proves to be daunting, since there is too much fluttering. For example a servo might move from 0º to 180º in 3 seconds but returning to 0º will take 5 minutes before start to return to 0º. This is later found to be Pi problem where there is too much interruption such as using CMUcam4 at the same time, and using programmer in Pi and using PicoBorg all at the same time.

Several libraries are tested, such as WiringPi libraries, DMA libraries, and others, but none of it gives required movement of servo. Therefore the setup of PWM is considered not successful. The only successful setup is mentioned in Section 4.4. The following shows several testing of PWM on LED using WiringPi libraries and adapted to servo.

```
#include
<stdio.h>
#include <wiringPi.h>

// LED Pin - wiringPi pin 0 is BCM_GPIO 17.
// Testing LED on RPi.
#define  LED      0

int main (void)
{
  printf ("Raspberry Pi blink\n") ;

  if (wiringPiSetup () == -1)
    return 1 ;

  pinMode (LED, OUTPUT) ;

  for (;;)
  {
    digitalWrite (LED, 1) ;       // On
    delay (500) ;         // mS
    digitalWrite (LED, 0) ;       // Off
    delay (500) ;
  }
  return 0 ;
}
```

The code above works since LED can be used in 3.3V environment. Tweaking the code to suits the servo, does not work since it requires 4.8V to 6V. Acquisition of the PicoBorg helps to shift the logic level as shown in the image below, where yellow line is the input (3.3V) and blue line (5V) is the output.

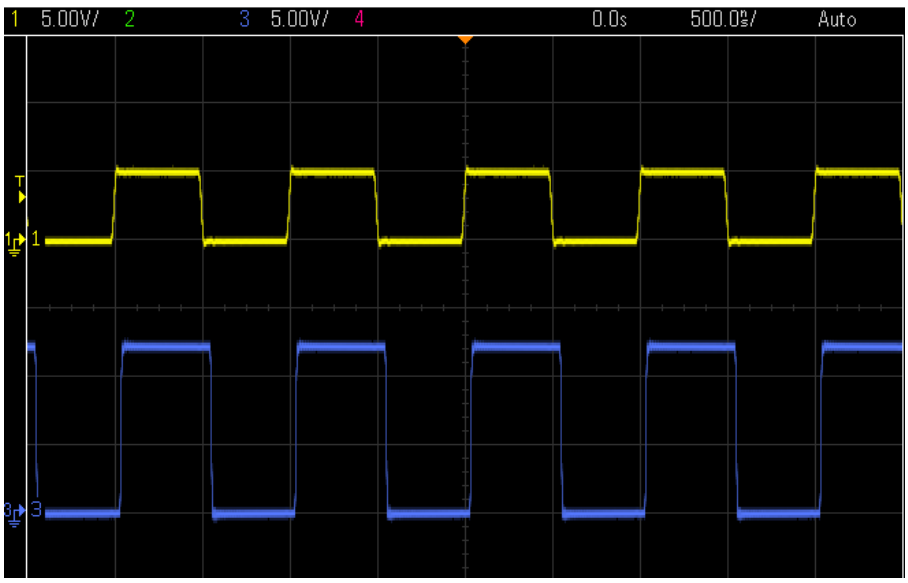

Figure 13: 3.3V to 5V logic level

```
import wiringpi2 as wiringpi
wiringpi.wiringPiSetup()
M1 = 7
M2 = 1
wiringpi.softPwmCreate(M1, 0, 100)
wiringpi.pinMode(M2, 2)
wiringpi.softPwmWrite(M1,50)      # 0 to 100
wiringpi.pwmWrite(M2, 512)        # 0 to 1024, 512 is central
```

The code above shows how we used WiringPi libraries to control the servo in Python. SoftPWMCreate is one of the ways to let us control more than just one PWM output, since we are using Pan/Tilt Servo, we needed two PWM outputs. So, creating the Software PWM and Hardware PWM allows us to control two servos.

The following connection is required since PicoBorg actually just drives motor, not the servo. The weak pull-up resistor (about 10k) is required.

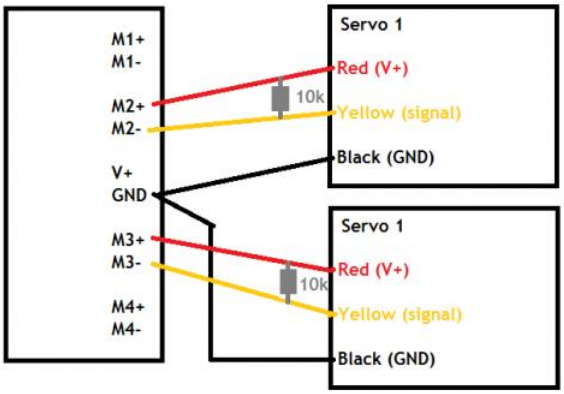

Figure 14: Servo Connection

4.2 Colour bounding for Red, Green and Blue Object

The CMUcam4GUI have two purposes, to show that it is working and to bound color for color tracking purposes. The following shows the flowchart of properly bounding the color according to the creator of CMUcam4, Kwabena Agyeman from Carnegie-Mellon University.

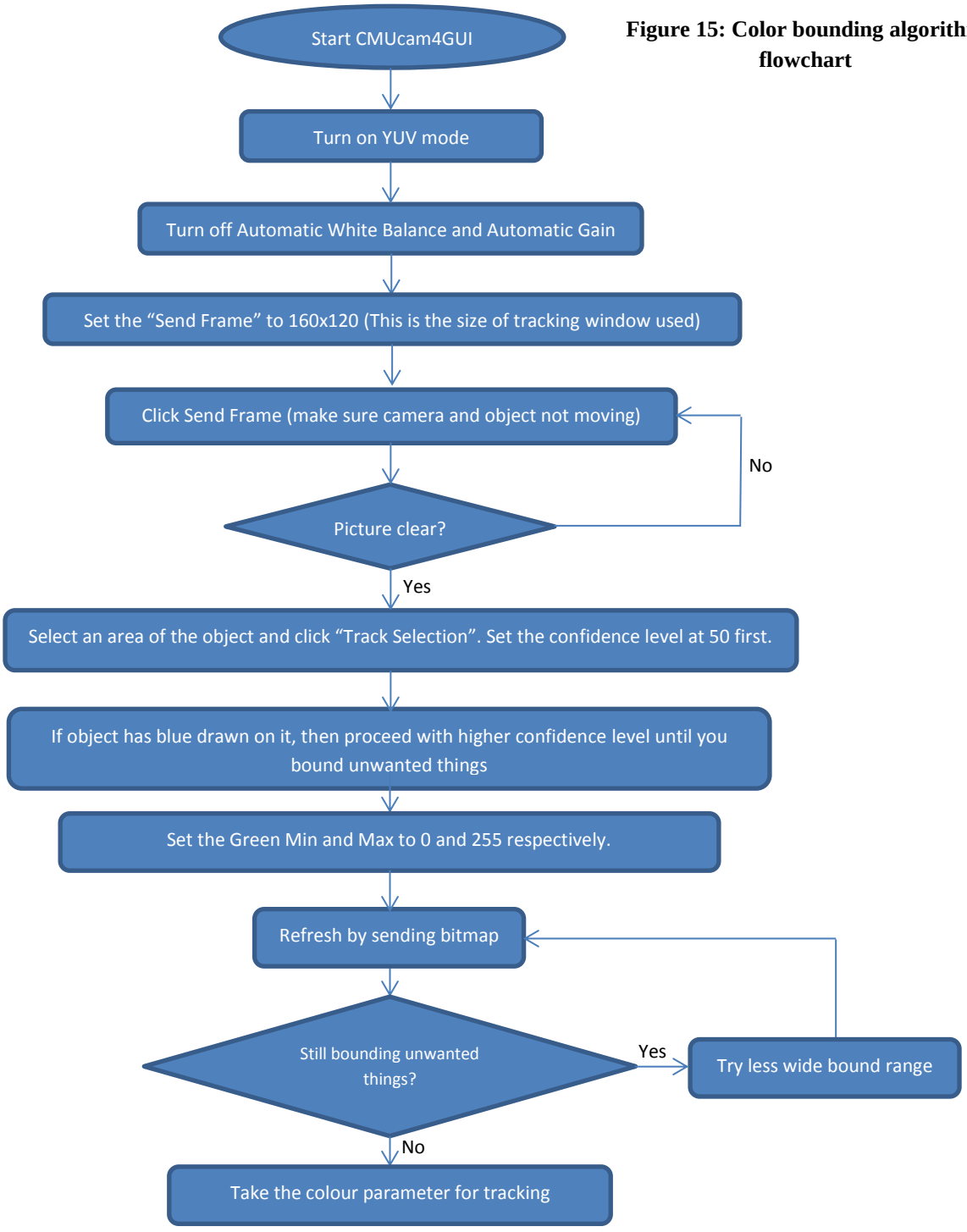

Figure 15: Color bounding algorithm flowchart

The following shows the color parameters taken for a Red, Green and Blue Keychain.

Colour	RMin	RMax	GMin	GMax	BMin	BMax
Blue	**76**	**94**	**113**	**129**	**188**	**207**
Green	**66**	**95**	**86**	**100**	**66**	**90**
Red	**186**	**253**	**0**	**90**	**66**	**121**

Table 2: Color parameter (subject to lighting environment)

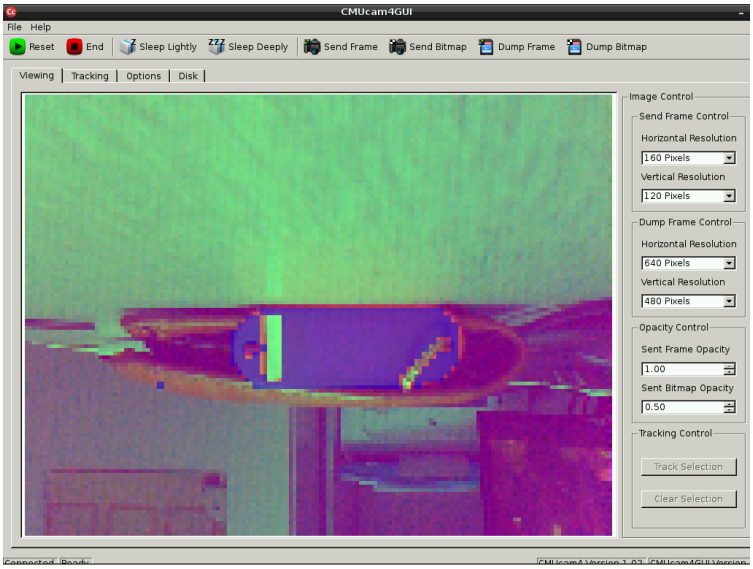

Figure 16: Blue tracked pixel of a red keychain

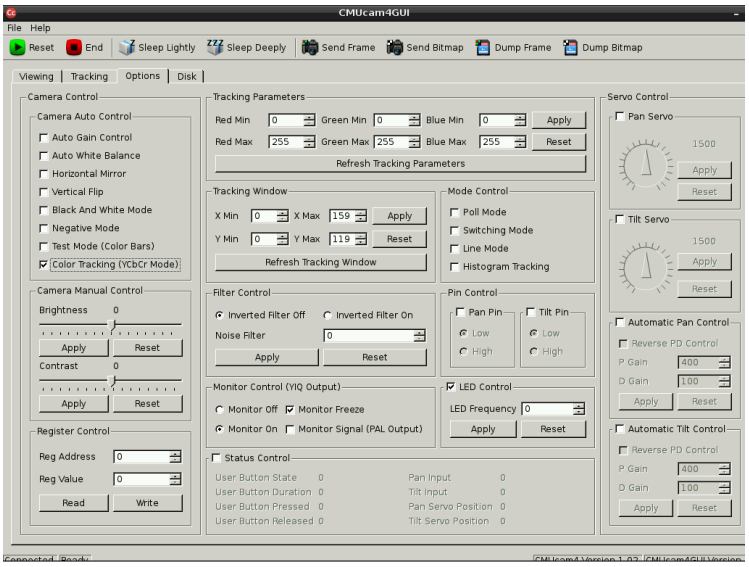

Figure 17: Options available on CMUcam4GUI

4.3 Interfacing RPi and CMUcam4 using Python

The reason for using Python is that it is higher level language compared to C and it is easier to pass the instructions to CMUcam4 using python compared to the C language. The following code snippet shows several initialization of the CMUcam4 before start tracking the object.

```
import threading
import serial
import time
import RPi.GPIO as GPIO
import re

cam = serial.Serial("/dev/ttyUSB0", baudrate=19200, timeout=0)
cam.bytesize = serial.EIGHTBITS
cam.parity = serial.PARITY_NONE
cam.stopbits = serial.STOPBITS_ONE
cam.xonxoff = False
cam.rtscts = False
cam.dsrdtr = False
cam.writeTimeout = 2

cam.write("GV"+chr(13))
ACK1 = cam.read(18)
print repr(ACK1)
time.sleep(3)
print "Testing LED for successful communication"
time.sleep(2)
print "now LED blinking at 10Hz"
cam.write("L1 10"+chr(13))
time.sleep(2)
print "Setting back LED to 0Hz"
cam.write("L1 0"+chr(13))
time.sleep(2)
print "Turn Automatic Gain and Auto White Balance off"
cam.write("AG 0"+chr(13))
cam.write("AW 0"+chr(13))
time.sleep(1)
print "Tracking color mode is set to YUV"
cam.write("CT 1"+chr(13))
time.sleep(1)
```

The following snippet shows how the extraction of the mx and my value value to be feed into servo.

```
while True:
response = cam.read(60)
if response.startswith('T'):
        print response
        val = response.split()
        print val[1]        #mx value
        print val[2]        #my value
        time.sleep(0.033333333333333)
        cam.close()
```

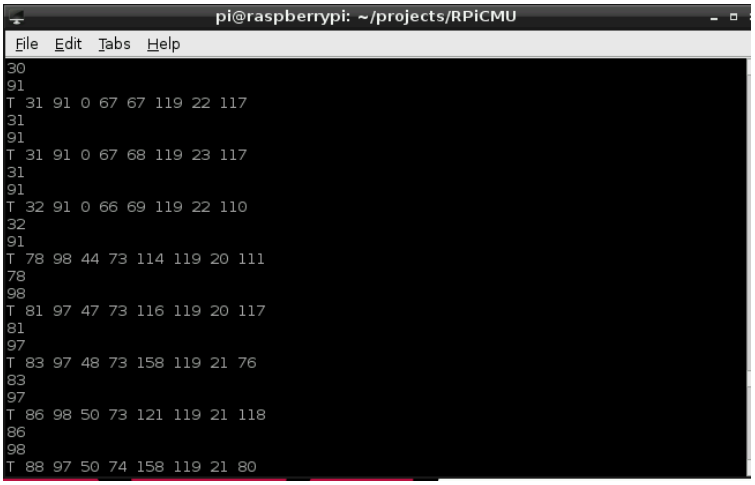

Figure 18: T data packet and extraction of mx and my value

4.4 Servo Positioning

Since the pan/tilt servo requires two PWM (Pulse Width Modulation) for its operation and only one hardware PWM is generated by the Raspberry Pi, the focus is now on how to convert any of GPIOs into semi-hardware PWM. This is made possible thanks to WiringPi library that is available on the github.com. The following is C code to implement the PWM setting in both the hardware PWM GPIO (pin 18) and semi-hardware PWM GPIO (pin 17). The code is however implemented in C language, and therefore need to be done in Python language.

The current progress is that 0º, 90º and 180º are able to be implemented. This is done by setting a time for each degree such as 0.3ms for 0º and 2.1ms for 180º. However, it still fails to go to right or left in accordance to the CMUcam4 tracking data. This may be due to lack of real-time application or just some unfamiliarity with Python language and servo.

4.5 Problems Faced during Setup of Project

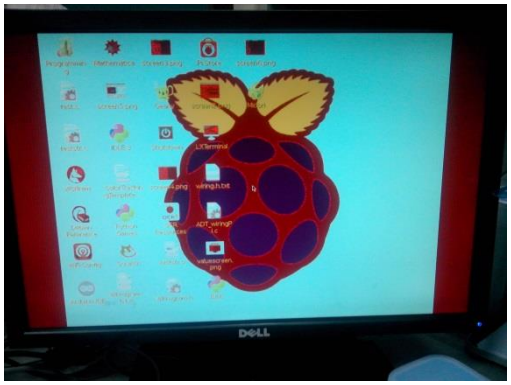

Figure 19: Weak signal of HDMI cause wrong display

Figure 16 shows the screen of a monitor that became red in colour. This means the HDMI signal from the Raspberry Pi does not possess enough power to display the colour correctly in large monitors (19inches). The solution is to use smaller monitor like 15in Samsung SyncMaster 153v model, or using HDMI boost option (but this can burns out Pi) or using better HDMI-VGA adapter.

Another problem is that the Raspberry Pi might not even boot into the home screen, i.e. rebooting many times. This is caused by corrupted SD card image of the operating system, or just incorrect setup, i.e. the monitor cable is not plugged in properly, and some component might take too much power from the Raspberry Pi until it shorts. The solution is to recheck every connection and make sure SD card is readable. Also remove any unwanted USB connection or other hardware attachment and try again. Usually the case that happens with the FYP's Raspberry Pi is, the hardware attached will short it. The hardware mentioned is the Wi-Fi dongle from D-Link and once removed the dongle, Raspberry Pi works as usual.

A more technical problem will be in terms of serial communication between Pi and CMUcam4. First try using only jumper from CMUcam4 UART to Pi UART. But this is not successful method. In the end, a virtual USB communication module is needed, so we used UART-USB Adaptor that helps to communicate and set up serial programming successfully.

CHAPTER 5

CONCLUSION

Following the result that is gathered, we are able to track an object. However, the servo still needs to be interfaced in order to confirm the workability of the project. The value of mx and my will help to determine the centroid positioning for servo later. Thus far, we are able interface the RPi and CMUcam4 successfully, able to track a red keychain and able to find the position of the keychain. However, the servo is not yet able to work with the rest of the system and this proves that more hardware is required to successfully conclude this project. The hope of this project in the future is to further develop the way to get the distance of the camera from the object and then develop specific action, i.e. grabbing the object. The other recommendation is to develop a way to predict the object movement based on the velocity of the object and specific distance to be fixed.

CHAPTER 6

REFERENCES

1. Wikipedia. *Computer*. 2014; Available from: http://en.wikipedia.org/wiki/Computer.
2. Mitchell, G., *The Raspberry Pi single-board computer will revolutionise computer science teaching [For & Against]*. Engineering & Technology, 2012. **7**(3): p. 26-26.
3. Severance, C., *Eben Upton: Raspberry Pi*. Computer, 2013. **46**(10): p. 14-16.
4. Andrews, C., *Easy as pi*. Engineering & Technology, 2013. **8**(3): p. 34-37.
5. Kavithaa, R., R.U. Babu, and C.R. Deepak. *Simple pendulum analysis — A vision based approach*. in *Computing, Communications and Networking Technologies (ICCCNT),2013 Fourth International Conference on*. 2013.
6. Lyons, K.R. and S.S. Joshi. *Paralyzed subject controls telepresence mobile robot using novel sEMG brain-computer interface: Case study*. in *Rehabilitation Robotics (ICORR), 2013 IEEE International Conference on*. 2013.
7. Kaifu, Y., et al. *Efficient Color Boundary Detection with Color-Opponent Mechanisms*. in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. 2013.
8. Chandan, B., et al. *Novel approach to lane and path detection in unmanned ground vehicles*. in *Advances in Technology and Engineering (ICATE), 2013 International Conference on*. 2013.
9. Hummel, S., et al., *A comparison of accuracy and cost of LiDAR versus stand exam data for landscape management on the Malheur National Forest.* Journal of forestry, 2011. **109**(5): p. 267-273.
10. Bikman, J.D., T.W. Meiswinkel, and J.M. Conrad. *A vehicle implementation of a color following system using the CMUcam3*. in *Southeastcon, 2009. SOUTHEASTCON '09. IEEE*. 2009.
11. Wan, E.A. and R. Van Der Merwe. *The unscented Kalman filter for nonlinear estimation*. in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*. 2000. IEEE.
12. Jain, R., R. Kasturi, and B.G. Schunck, *Machine vision*. Vol. 5. 1995: McGraw-Hill New York.

APPENDICES

APPENDIX I: Python Programming

tunedpytest.py

```
import threading
import serial
import time
import RPi.GPIO as GPIO
import re

cam = serial.Serial("/dev/ttyUSB0", baudrate=19200, timeout=0)
cam.bytesize = serial.EIGHTBITS
cam.parity = serial.PARITY_NONE
cam.stopbits = serial.STOPBITS_ONE
cam.xonxoff = False
cam.rtscts = False
cam.dsrdtr = False
cam.writeTimeout = 2

cam.write("GV"+chr(13))
ACK1 = cam.read(18)
print repr(ACK1)
time.sleep(3)

print "Testing LED for successful communication"
time.sleep(2)
print "now LED blinking at 10Hz"
cam.write("L1 10"+chr(13))
time.sleep(2)
print "Setting back LED to 0Hz"
cam.write("L1 0"+chr(13))
time.sleep(2)
print "Turn Automatic Gain and Auto White Balance off"
cam.write("AG 0"+chr(13))
cam.write("AW 0"+chr(13))
time.sleep(1)
print "Tracking color mode is set to YUV"
cam.write("CT 1"+chr(13))
time.sleep(1)
print "Resetting tracking parameter for tracking a red keychain"
cam.write("ST 186 253 0 90 66 121"+chr(13))
time.sleep(1)
ACK = cam.read(60)
print repr(ACK)
time.sleep(1)
print "Tracking a red keychain"
cam.write("TC"+chr(13))
time.sleep(0.3)
```

```
while True:
response = cam.read(60)
if response.startswith('T'):
        print response
        val = response.split()
        print val[1]        #mx value
        print val[2]        #my value
```

APPENDIX II: CMUcam4 Command List (cmucam.org)

The serial communication parameters are as follows:

- 19,200 Baud Rate – See "BM" (Baud Mode) command to change after startup
- 1 Start Bit
- 8 Data Bits
- 1 Stop Bit – See "DM" (Delay Mode) command to change after startup
- No Parity/Sticky Bit
- No Flow Control – software (Xon (ASCII 17) / Xoff (ASCII 19)) or hardware (RTS/CTS)

- '\t' (ASCII 9 – tab) is converted to ' ' (ASCII 32 – space)
- '\b' (ASCII 8 – backspace) is evaluated by deleting the last character received
- Lower case characters ('a' through 'z') are converted to upper case characters ('A' through 'Z')
- ASCII characters 0 – 7, 10 – 12, 14 – 31, 127 – 255 are ignored (thrown away)

Functionally Grouped Command Listing

- System Level Commands
  - "GV" Get Version 25
  - "RS" Reset System 40
  - "SD" Sleep Deeply 42
  - "SL" Sleep Lightly 44
- Camera Module Commands
  - "CB" Camera Brightness 10
  - "CC" Camera Contrast 10
  - "CR" Camera Register Read 12
  - "CW" Camera Register Write 13
- Camera Sensor Auto Control Commands
  - "AG" Auto Gain Control 6
  - "AW" Auto White Balance 7
- Camera Format Commands
  - "HM" Horizontal Mirror 26
  - "VF" Vertical Flip 55
- Camera Effect Commands
  - "BW" Black and White Mode 8
  - "NG" Negative Mode 35
- Auxiliary I/O Commands
  - "GB" Get Button State 20
  - "GD" Get Button Duration 20
  - "GP" Get Button Pressed 23
  - "GR" Get Button Released 24
  - "PI" Pan Input 36
  - "PO" Pan Output 38
  - "TI" Tilt Input 50
  - "TO" Tilt Output 51
  - "GI" Get Inputs 22
  - "SO" Set Outputs 45
  - "L0" LED Off (0) 28
  - "L1" LED On (1) 29
- Servo Commands
  - "GS" Get Servo Position 24
  - "SS" Set Servo Position 46
  - "AP" Automatic Pan 6
  - "AT" Automatic Tilt 7
  - "PP" Auto Pan Parameters 38
  - "TP" Auto Tilt Parameters 52
- Data Rate Commands
  - "BM" Baud Mode 8
  - "DM" Delay Mode 17
- Television Commands
  - "M0" Monitor Off (0) 31
  - "M1" Monitor On (1) 31
  - "MF" Monitor Freeze 32
  - "MS" Monitor Signal 33
- Color Tracking Commands
  - "GT" Get Tracking Params. 25
  - "GW" Get Tracking Window 26
  - "ST" Set Tracking Params. 47
  - "SW" Set Tracking Window 48
  - "TC" Track Color 49
  - "TW" Track Window 52
  - "GH" Get Histogram 21
  - "GM" Get Mean 22
  - "PM" Poll Mode 37
  - "LM" Line Mode 29
  - "SM" Switching Mode 44
  - "TM" Test Mode 51
  - "CT" Color Tracking 12
  - "HT" Histogram Tracking 27
  - "IF" Inverted Filter 28
  - "NF" Noise Filter 35
- File System Commands
  - "CA" Change Attributes 9
  - "CD" Change Directory 11
  - "DI" Disk Information 16
  - "DS" Disk Space 18
  - "FM" Format Disk 19
  - "LS" List Directory 30
  - "MK" Make Directory 33
  - "MV" Move Entry 34
  - "PL" Print Line 36
  - "PR" File Print 39
  - "RM" Remove Entry 40
  - "UM" Unmount Disk 54
- Image Capture Commands
  - "DB" Dump Bitmap 13
  - "DF" Dump Frame 14
  - "SB" Send Bitmap 41
  - "SF" Send Frame 42
- Idle Command 27
- Invalid Command 27

Track Color TC [*red min – Integer*] [*red max – Integer*] [*green min – Integer*] [*green max – Integer*] [*blue min – Integer*] [*blue max – Integer*] '\r'