Lab 3 Report

Jinzhi Cai

March 8, 2020

Contents

1 Introduction

2 Image on Raspberry Pi 3B

2.1 Setup

The installation of Raspberry Pi 3B require following components.

- Raspberry PI 3B
- SD card
- Laptop
- Raspberry Pi OS Image

The image can be download in the official website of raspberry pi website.

2.2 Procedure

To install the Raspberry Pi system to the SD card require a image burning software. After download the software and the image of the OS. Insert the SD card and start the process. After 20 min, the system will be install into the sd card. Putting the sd card to the Raspberry Pi 3B and power the system. It will boot to the system.

2.3 Result

After install, it will display a Raspberry Desktop interface.

3 Connect to servo motor driver using i2c utility & Control Servo motor using i2c utility &Control Pan-tilt using c++ class

3.1 Setup

In order to make the Raspberry Pi to control the I2C control board and the servo attach with it, following items is require.

- Raspberry PI 3B
- SD card
- PCA9685
- Power Supply
- PTU
- Analog Discovery

3.2 Procedure

There are three processes to finish the goal.

3.2.1 Enable the I2C Port in the Raspberry Pi

The process to enable the I2C port in Raspberry Pi is following.

1) Open Raspberry Pi command line.

2) Run command **sudo rasp-config**

3) Open **Interface** option and open **I2C Interface**

4) reboot Raspberry Pi

To testing the is the I2C working, type in command **gpio readall**, **ls /dev/*i2c***, and **gpio i2cdetect** to show i2c device and check all the I2C slave in the bus.

3.2.2 Adapt the Adafruit library to WiringPi environment

Because the I2C board is manufactured by Adafruit, it provided a library to drive the PWM board. However, it only work on *Arduino* platform. Therefore, it require some change in order to access the I2C device in the raspberry.

After reading the library Adafruit provided, the structure of this program is very clear.

uint8_t read8(uint8_t addr); void write8(uint8_t addr, uint8_t d);
Adafruit_PWMServoDriver(); void begin(uint8_t prescale = 0);

All the function above is require to use the **wire.h** and **Arduino.h** and that need to be change to the WiringPi environment.

```
Adafruit_PWMServoDriver::Adafruit_PWMServoDriver(){
        _i2caddr=0x70;
}

void Adafruit_PWMServoDriver::begin(uint8_t prescale) {
        _i2c=wiringPiI2CSetup(_i2caddr);
        reset();
```

```
        if (prescale) {
                setExtClk(prescale);
        } else {
        // set a default frequency
                setPWMFreq(1000);
        }
        // set the default internal frequency
        setOscillatorFrequency(FREQUENCY_OSCILLATOR);
}

uint8_t Adafruit_PWMServoDriver::read8(uint8_t addr) {
        return wiringPiI2CReadReg8(_i2c,addr);
}

void Adafruit_PWMServoDriver::write8(uint8_t addr, uint8_t d) {
        wiringPiI2CWriteReg8 (_i2c, addr, d);

}
```

After apply those change and isolate the **wire.h** and **Arduino.h**, the program should be able to run in the Raspberry Pi environment.

3.2.3 Testing circuit by attach servo and power supply

To test the program follow process is required.

1) Set up the power supply and set the output to **5V**

2) Connect the power to the **V+** and **GND** port in PCA9685.

3) Connect Raspberry Pi 3B 5v power to VCC and GND port in PCA9685.

4) Connect Raspberry Pi 3B SCL, SDA to the same port in PCA9685.

5) Set up Pull-Up resistor to SCL SDA and the VCC line.

6) Connect Analog Discovery DIO0 and DIO1 to the SCL SDA port.

After set up the hardware, use **gpio i2cdetect** to discovery the PCA9685 and the address of the PCA9685 is **0x40**.

To control the servo, program should have a *Adafruit_PWMServoDriver* object at the memory location **0x40**. Then, program will set up the frequency for the oscillator and update. In the end, the driver will set on the pwm frequency for each port.

```
#define SERVO_FREQ 50
        int main(){
                Adafruit_PWMServoDriver pwm(0x40);
                pwm.begin();
                pwm.setOscillatorFrequency(27000000);
                pwm.setPWMFreq(SERVO_FREQ);
                int value=0;

                pwm.setPWM(0, 0, 350);
                pwm.setPWM(1, 0, 350);
                while(1){
                        cin >> value;
                        pwm.setPWM(0, 0, value);
```

```
            }
        }
```

This program will set up the pwm driver and apply pwm signal to the port. In the while loops, system will read the user input and use to adjust the servo.

After testing the servo angle, controlling the Pan-tilt unit can be done by applying the correct pwm signal.

3.3 Result

By using **gpio i2cdetect**, the address for the servo driver is 0x40. By using the angle testing program, the pwm range is from 250 to 420. After using apply all the program into the system, the pan-tilt will point to the target input into the system.

4 Connect your phone to RPi using Bluetooth

4.1 Setup

In order to make the Raspberry Pi to control the Bluetooth, following items is require.

- Raspberry PI 3B
- SD card
- Analog Discovery

4.2 Procedure

The process to enable the Bluetooth port in Raspberry Pi is following.

1) Open Raspberry Pi command line.
2) Run command **sudo rasp-config**
3) Open **Interface** option and open **Bluetooth Interface**
4) reboot Raspberry Pi
5) Run command **sudo bluetoothctl** to start the bluetoothctl
6) Run command in bluetoothctl **power on** to enable the power setting
7) Run command in bluetoothctl **agent on** to start the agent
8) Run command in bluetoothctl **discoverable on** to allow disable
9) Run command in bluetoothctl **default-agent** to use the default agent
10) Use phone to find the raspberry-pi bluetooth device.
11) Confirm the pair request and trust the device.
12) Run following command in command line

    hciconfig hci0 piscan

    sdptool add SP

13) Run command **sudo apt-get install libbluetooth-dev**

The following program is provide by TA. By running this program, phone which connect to the raspberry will receive a message from the Raspberry Pi.

```
#include <stdio.h>
#include <unistd.h>
#include <sys/socket.h>
#include "bluetooth.h"
#include "rfcomm.h"

using namespace std;
int main(int argc, char **argv)
{
        struct sockaddr_rc loc_addr = { 0 }, rem_addr = { 0 };
        char buf[1024] = { 0 };
        int s, client, bytes_read;
```

```
        socklen_t opt = sizeof(rem_addr);

        // allocate socket
        s = socket(AF_BLUETOOTH, SOCK_STREAM, BTPROTO_RFCOMM);

        // bind socket to port 1 of the first available
        // local bluetooth adapter
        loc_addr.rc_family = AF_BLUETOOTH;
        //loc_addr.rc_bdaddr = *BDADDR_ANY;
        loc_addr.rc_channel = (uint8_t) 1;
        bind(s, (struct sockaddr *)&loc_addr, sizeof(loc_addr));

        // put socket into listening mode
        listen(s, 1);

        // accept one connection
        client = accept(s, (struct sockaddr *)&rem_addr, &opt);

        ba2str( &rem_addr.rc_bdaddr, buf );
        fprintf(stderr, "accepted connection from %s\n", buf);
        memset(buf, 0, sizeof(buf));

        // read data from the client
        bytes_read = read(client, buf, sizeof(buf));
        if( bytes_read > 0 ) {
                printf("received [%s]\n", buf);
        }

        // close connection
        close(client);
        close(s);
        return 0;
}
```

4.3 Result

After follow the procedure above, the phone successfully connect to the Raspberry Pi and receive the default message.

5 Send data to RPi using a BT class and app & Control PTU using your phone

5.1 Setup

In order to make the Raspberry Pi to control the I2C control board and the servo attach with it by bluetooth, following items is require.

- Raspberry PI 3B
- SD card
- PCA9685
- Power Supply
- Analog Discovery
- PTU

5.2 Procedure

The process of allow connection between the pwm servo and the bluetooth take three major steps.

5.2.1 Program the bluetooth class

By reading the program, the structure can be discovery. The way program and bluetooth communication by file describer. It indicate it is possible to send the data out by use the **write** function call.

```
rfcomm_server& operator<<(std::string s){
            write(client,s.c_str(),s.size());
            write(client,"\n",1);
            return *this;
    }
    rfcomm_server& operator>>(std::string* s){
            char buf[1024] = { 0 };
            int bytes_read=-1;
            while (bytes_read==-1){
                    memset(buf, 0, sizeof(buf));
                    bytes_read= read(client, buf, sizeof(buf));
            }
            s->append(buf);
            return *this;
    }
```

5.2.2 Connect the bluetooth and send out a message

By using the previous created class, a small program can be created to test the class and the ability to transmit message. Statement **rfcomm_server rout** and **rout« "good news"**.

5.2.3 Connect the bluetooth and pwm servo

By using the previous created classes, it will be easy to create a program to use bluetooth to control the servo. The program contain two parts. The first part is to setup the I2C connection and the bluetooth connection. The second part is read the message that came from the bluetooth and use this value to update pwm signal.

5.3 Result

When testing the bluetooth, the phone will receive a **good news** message from the raspberry pi. When using the bluetooth to control the servo, the servo will move to pre-programmed location base on the command phone send out.

6 Get image using camera and raspistill

6.1 Setup

In order to make the Raspberry Pi to get image using camera and raspistill, following items is require.

- Raspberry PI 3B
- SD card
- Raspberry Camera

6.2 Procedure

Install the camera of the Raspberry Pi require the Raspberry Pi to power off during the installation of the camera. Becasuse the sensor in the camera is very sensitive, installing the camera with power will cause a power pulse on the camera sensor and destroy the sensor in the camera. After start the Raspberry with the camera. Following steps is required to start the camera.

1. Run following command in the command line **sudo rasp-config**
2. In the config pag, open the **interface** page and turn on the **camera** option.
3. After exist the config page, reboot the system.
4. After finish reboot, run command in the command line **raspistill -o text.jpg**

6.3 Result

After run thought all command above, a image will be in the home directory call ”text.jpg”.

7 Install OpenCV, check with -version & Get image using OpenCV & Face recognition using OpenCV

7.1 Setup

In order to make the Raspberry Pi to get image and face recognition using OpenCV, following items is require.

- Raspberry PI 3B
- SD card
- Raspberry Camera

7.2 Procedure

To finish this task take three steps.

7.2.1 Install OpenCV and create program

To install the openCV, run following command **sudo apt-get install libopencv-dev**.

The program take three parts.
The first part is to get the video device from the system.

```
VideoCapture cap;
// open the default camera,
 use something different from 0 otherwise;
// Check VideoCapture documentation.
if(!cap.open(0))
        return 0;
```

The second part is to get image frame from the video device.

```
Mat frame;
cap >> frame;
if( frame.empty() )
        continue; // end of video stream
```

The third part is to put the image to the face recognition and send out the face location.

```
FaceLoc f=detectAndDisplay(frame);
```

Because the speed of the Raspberry Pi, some change in the face recognition program required to improve the perform.
face_cascade.detectMultiScale(frame_gray, faces, 1.1, 2, 0|CV_HAAR_SCALE_IMAGE, Size(100, 100));
Decreasing the size of face recognition will improve the speed.

```
FaceLoc detectAndDisplay( Mat frame ){
        std::vector<Rect> faces;
        Mat frame_gray;
        int frame_y= frame.rows;
        int frame_x= frame.cols;
        // cout <<endl;
        cvtColor( frame, frame_gray, CV_BGR2GRAY );
        equalizeHist( frame_gray, frame_gray );
```

```
        //-- Detect faces
        face_cascade.detectMultiScale( frame_gray, faces, 1.1, 2,
         0|CV_HAAR_SCALE_IMAGE, Size(100, 100) );
        Point center( frame_x*0.5, frame_y*0.5 );
        ellipse( frame, center, Size( 20, 20), 0, 0, 360,
        Scalar( 255, 0, 255 ), 4, 8, 0 );
        FaceLoc f;
        f.x=0;
        f.y=0;
        for( size_t i = 0; i < faces.size(); i++ )
        {
                Point center( faces[i].x + faces[i].width*0.5,
                 faces[i].y + faces[i].height*0.5 );
                ellipse( frame, center, Size( faces[i].width*0.5,
                 faces[i].height*0.5), 0, 0,
                  360, Scalar( 255, 0, 255 ), 4, 8, 0 );
                f.x=faces[0].x+ faces[0].width*0.5-(frame_x*0.5);
                f.y=faces[0].y + faces[0].height*0.5-(frame_y*0.5);
        }
        imshow("this is you, smile! :)", frame);
        waitKey(1);
        return f;
}
```

7.2.2 Testing camera

The program to test the program is following.

```
        cap >> frame;
        Mat frame;
        imshow("this is you, smile! :)", frame);
        waitKey(10);
```

The **waitKey(10)** is required because the speed of the cpu is too faster. CPU have to wait for the camera to get the image.

7.2.3 Testing face recognition

The program to test the face recognition is following.

```
        FaceLoc f=detectAndDisplay(frame);
```

The function for analysis the frame is following.

```
FaceLoc detectAndDisplay( Mat frame ){
        std::vector<Rect> faces;
        Mat frame_gray;
        cvtColor( frame, frame_gray, CV_BGR2GRAY );
        equalizeHist( frame_gray, frame_gray );

        //-- Detect faces
        face_cascade.detectMultiScale( frame_gray, faces, 1.1, 2,
        0|CV_HAAR_SCALE_IMAGE, Size(100, 100) );
        Point center( frame_x*0.5, frame_y*0.5 );
        ellipse( frame, center, Size( 20, 20), 0, 0, 360,
        Scalar( 255, 0, 255 ), 4, 8, 0 );
        FaceLoc f;
```

```
        f.x=0;
        f.y=0;
        for( size_t i = 0; i < faces.size(); i++ ){
                Point center( faces[i].x + faces[i].width*0.5,
                faces[i].y + faces[i].height*0.5 );
                ellipse( frame, center, Size( faces[i].width*0.5,
                faces[i].height*0.5), 0, 0, 360,
                Scalar( 255, 0, 255 ), 4, 8, 0 );
        }
        imshow("this is you, smile! :)", frame);
        waitKey(1);
        return f;
}
```

7.3 Result

After use the camera test program, one windows will appear and a video stream will display. After load the face recognition program, the faces in the video will be marked by a red cycle around it.

8 Track Face, keep face centered & Add BT command: PTU control + Track On/Off

8.1 Setup

In order to make the Raspberry Pi to get image and face recognition using OpenCV, following items is require.

- Raspberry PI 3B
- SD card
- Raspberry Camera
- PCA9685
- Power Supply
- Analog Discovery
- PTU

8.2 Procedure

To finish this task, the all the program that use in the previous lab is required. The program take three parts.

8.2.1 PWM controller

The PWM controller is use the **Adafruit_PWMServoDriver**. The PTU require two servo. It require two pwm port to operate.

```
pwm.begin();
pwm.setOscillatorFrequency(27000000);
pwm.setPWMFreq(SERVO_FREQ);

......

if (f.x>0)h--;
if (f.x<0)h++;
if (f.y>0)v++;
if (f.y<0)v--;

if (h>turn[0])h=turn[0];
if (h<turn[2])h=turn[2];
if (v>rise[0])v=rise[0];
if (v<rise[2])v=rise[0];

pwm.setPWM(0, 0, h);
pwm.setPWM(1, 0, v);
```

8.2.2 Bluetooth controller

The bluetooth controller is a little bit different. Because the read is block method, it is impossible to run pwm and listening bluetooth message in the same time. Therefore, multi-thread is required in the program.

```
void readInput(){
        while (1){
                string goodone=rout.readRF();
                if ((goodone)[0]=='M'){
                        cout << "TP3" << endl;
                        on=0;
                }
                if ((goodone)[0]=='L'){
                        cout << "TP3" << endl;
                        on=1;
                }
        }
}
```

The function above is use to receive the message from the bluetooth that send by phone. It will be running in another thread.

thread t(readInput);

8.2.3 Camera and Face Recognition Handler

The camera and face recognition handler is use to in take in the image flow from the camera and run a face recognition process on the frame.

```
Mat frame;
cap >> frame;
if( frame.empty() )
        continue; // end of video stream
FaceLoc f=detectAndDisplay(frame);
```

By using the statement above, the image from camera will be send to face recognition process and the location of the face in the frame. Those information will be use to adjust the PTU to make it point to the correct location of the face.

8.3 Result

After load all the program to Raspberry Pi, camera will looking for the face appear in the frame and control the PTU to make the camera point to the center of the face it detected.

9 Conclusion