Group 6: Giovanni Legname + Tieren Peng + Tien Le + Luong Nguyen

PiBooth

Helsinki Metropolia University of Applied Sciences

Devices project report

9/5/2016

Contents

1 Original plans

Our original idea for the project is to make a music player based on the surrounding environment.

Data collected from the temperature sensor, light sensor and motion sensor will be analyzed to decide which suitable genre of music that the Raspberry Pi should play. For example upbeat music is nice to be played in a summer morning, where as a romantic song can cozy up a snowy winter night. Or if in the room there are a lot of motions, pi will play some active songs and if no motions, pi will automatically lower and then turn off the music.

The information of the song which is playing will then be displayed on the lcd screen for the user. The songs will come from internet sources such as Soundcloud, Spotify or from the SD card of the Raspberry Pi.

2 Project progress

Week 1: In the first week we started to learn programming in Python on Viope. We discussed and choose the first idea of the project. We started the project by installing the operating system on our Raspberry Pi and tried to run code on the PiBoard with the traffic light exercise.

Week 2: We ordered and received all the sensors and devices needed for the project. We tried to read data collected from the temperature sensor and light sensor on the PiBoard. It was more challenging than we expected since it took several tasks to convert the digital data received to actual temperature value in degree Celsius and light levels in Lux.

Week 3: This week we set up the motion sensor, displayed the data collected from all the sensors on the lcd screen and then sent the data string from the PiBoard to the Raspberry Pi every second.

Week 4: Since we noticed other teams also required speakers to implemented some kinds of music player to their project, and we had not started implementing our music player, we decided to change the project idea. We pivoted to making a photo booth, using the Raspberry Pi Camera to take portrait photo of users when there are any motions detected by the motion sensor. The Lux values are used to adjust the ISO of the

camera and the temperature recorded will be displayed on the output picture. This week we started working on taking picture using the Pi Camera and editing the photo taken.
Week 5: We continued adjusting the output picture and figure out a way to display the picture to the users. We choose to make a simple website which could display the latest photo taken. It took us a morning trying to install Meteor framework, which we already had previous experiences, but unfortunately the Raspberry Pi was not supported. We then install the Apache web server and successfully launched a website.
Week 6: We moved on to work on the database. By following the RRD tool instructions, we could save the temperature and light values into a database and automatically update the data. We generated a graph of the temperature and light values and put it in a bootstrap carousel on the front page of our website. We also used Twitter API to automatically update a status with the latest photo taken. Now we can see all the pictures we had on a Twitter page, not just the latest photo on the website.
Week 7: In the last week we tried to implement one last feature - face detection. Using API from kairos we hope to detect the gender of the user whose photo was taken. It turned out quite challenging and we are still not able to do it, but we strongly believe it can become a wow factor for our project, so we will try our best to have it working next week. In the last lab we made the booth from a carton box.The user interface of the PiBooth is a camera on the top, a motion sensor and a lcd screen to display instructions for users. Every other things will be hide behind the booth in order not to distract users. We also had a lot of fun decorating the booth and hope it would be a nice surprise for everybody. Finally we prepared the project report and poster.

3 Implementation details

3.1 Temperature sensor

PyBoard uses KTY81/210 temperature sensor. The temperature sensors in the KTY81 series have a positive temperature coefficient of resistance and are suitable for use in measurement and control systems.
To get the temperature information, the sensor should be declared by class ADC – analog to digital conversion by using command:

```
adc = pyb.ADC(pyb.Pin.board.X1)
```

After declaration, an output information can be read on the analog pin from the ADC by using command:

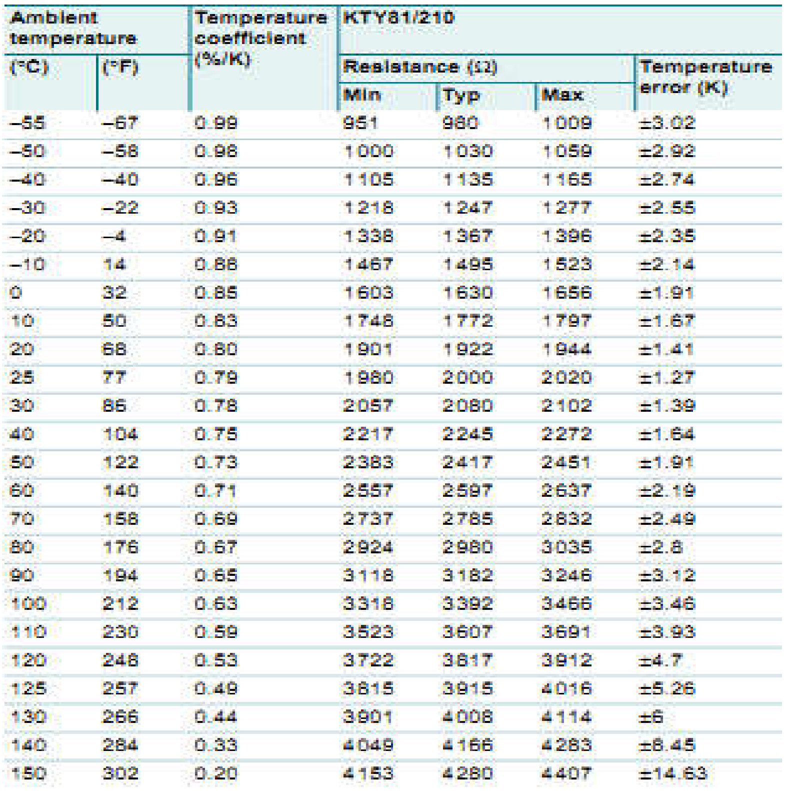
```
val = adc.read()
```

This output information needs to be calculated to become realizable information as resistance. Based on available formula, the resistance data can be calculated by 2 methods:

```
def calculateVout(adcVal):
       return 3.3 / 4095 * adcVal

def calculateRt(vOut):
       r = 1.5 * 10**3
       return vOut * r / (3.3 - vOut)
```

The resistance data is necessary to print out the temperature information by using a datasheets.



Ambient temperature		Temperature coefficient (%/K)	KTY81/210			
			Resistance (Ω)			Temperature error (K)
(°C)	(°F)		Min	Typ	Max	
−55	−67	0.99	951	980	1009	±3.02
−50	−58	0.98	1000	1030	1059	±2.92
−40	−40	0.96	1105	1135	1165	±2.74
−30	−22	0.93	1218	1247	1277	±2.55
−20	−4	0.91	1338	1367	1396	±2.35
−10	14	0.88	1467	1495	1523	±2.14
0	32	0.85	1603	1630	1656	±1.91
10	50	0.83	1748	1772	1797	±1.67
20	68	0.80	1901	1922	1944	±1.41
25	77	0.79	1980	2000	2020	±1.27
30	86	0.78	2057	2080	2102	±1.39
40	104	0.75	2217	2245	2272	±1.64
50	122	0.73	2383	2417	2451	±1.91
60	140	0.71	2557	2597	2637	±2.19
70	158	0.69	2737	2785	2832	±2.49
80	176	0.67	2924	2980	3035	±2.8
90	194	0.65	3118	3182	3246	±3.12
100	212	0.63	3318	3392	3466	±3.46
110	230	0.59	3523	3607	3691	±3.93
120	248	0.53	3722	3817	3912	±4.7
125	257	0.49	3815	3915	4016	±5.26
130	266	0.44	3901	4008	4114	±6
140	284	0.33	4049	4166	4283	±8.45
150	302	0.20	4153	4280	4407	±14.63

The range of normal Earth temperature is from -30^{0}C to 50^{0}C, assume that each 10^{0}C is a linear, the temperature can be printed out by method:

```
def calculateT(rt):
   tList = [-30,-20,-10,0,10,20,30,40,50]
   rtList = [1247,1367,1495,1630,1772,1922,2080,2245,2417]
   for i in range(1,len(rtList)+1):
      if rt >= rtList[i-1] and rt <= rtList[i]:
      alpha = (tList[i] - tList[i-1])/(rtList[i] - rtList[i-1])
      t = tList[i-1] + (rt - rtList[i-1])*alpha
      return t
```

3.2 Light sensor

PyBoard uses TSL2250 light sensor. The TSL2250 light sensor is a digital-output light sensor with a two-wire, SMBus serial interface. It combines two photodiodes and a companding analog-to-digital converter (ADC) on a single CMOS integrated circuit to provide light measurements over an effective 12 bit dynamic range with a response similar to that of human eyes.

The TSL2550 is designed for use with broad wavelength light sources. One of the photodiodes (channel 0) is sensitive to visible and infrared light, while the second photodiode (channel 1) is sensitive primarily to infrared light. An integrating ADC converts the photodiode currents to channel 0 and channel 1 digital outputs. Channel 1 digital output is used to compensate for the effect of the infrared component of ambient light on channel 0 digital output. The ADC digital outputs of the two channels are used to obtain a value that approximates the human eye response in the commonly used unit of Lux.

Light sensor uses class I2C to communicate. I2C is a two-wire protocol for communicating between devices. At the physical level it consists of 2 wires: SCL and SDA, the clock and data lines respectively.

I2C objects are created attached to a specific bus. They can be initialised when created, or initialised later on:

```
lux = pyb.I2C(1, pyb.I2C.MASTER, baudrate = 20000)
      lux.send(0x43, 0x39)
      Channel_0 = lux.recv(1,0x39)
      lux.send(0x83, 0x39)
      Channel_1 = lux.recv(1,0x39)
```

Base on the formulas given by document of the TSL2550 light sensor and the data received from I2C, light level can be calculated by the method:

```
def calculateLux(Channel_0,Channel_1):
      # Calculate Chord number & Step Number
      chordMask = 0b01110000
      stepMask = 0b00001111
      chordNum_0 = (Channel_0[0] & chordMask) >> 4
      stepNum_0 = (Channel_0[0] & stepMask)
      chordNum_1 = (Channel_1[0] & chordMask) >> 4
      stepNum_1 = (Channel_1[0] & stepMask)

      # Calculate Chord Value and Step Value
      chordValue_0 = int(16.5 * ((2**chordNum_0) - 1))
      stepValue_0 = stepNum_0 * (2**chordNum_0)
      chordValue_1 = int(16.5 * ((2**chordNum_1) - 1))
      stepValue_1 = stepNum_1 * (2**chordNum_1)

      # Calculate Count Value
      ch0Counts = chordValue_0 + stepValue_0 * stepNum_0
      ch1Counts = chordValue_1 + stepValue_1 * stepNum_1

      # Calculate Light level
      if ch0Counts != 0:
            r = ch1Counts / ch0Counts
      lightLevel = ch0Counts * 0.46 * math.exp(-3.13*r)
      return lightLevel
```

3.3 LCD screen

PyBoard uses HD44780 as LCD screen. An available class written by Will Pimblett is to control using LCD module.

With the help of HD44780 class, the LCD screen can be controlled after a simple declaration using command:

```
lcd = HD44780(i2c)
```

The initial state of the LCD screen is to display “Wave to shoot!”. When constant motion is detected, the LCD screen will respectively display 'Ready !!!', '3', '2', '1', 'say Cheese!', 'Lovely!' and ‘Thank you!’ every second.

3.4 Motion sensor

Motion sensor uses PIN class to communicate. A pin is the basic object to control I/O pins. It has methods to set the mode of the pin (input, output, etc) and methods to get and set the digital logic level.

All Board Pins are predefined as pyb.Pin.board.Name and initialize by Mode and Pull. In this project Pin name is Y12, Mode is pin.IN and Pull is PULL_UP.

```
p_in = Pin('Y12', Pin.IN, Pin.PULL_UP)
```

Then the data from sensor can be received by method:

```
motionDetect = p_in.value()
```

which is 0 or 1 depend on the logic level of pin.

To avoid oversensitivity of the motion sensor, when we first receive the value “1” which means motion detected, we will delay for 3 seconds and then check the motion again. If the motion is constantly we will send data to the Raspberry Pi to trigger the Pi Camera. Otherwise no further actions are needed.

3.5 Communicate data

Data of current temperature, light values and motion detection are concatenated into a single, separated by a white space. Each second a new data string is sent from the PiBoard to the Raspberry Pi by UART.

```
data = (str(temp), str(light), str(motionDetect))
input = ' '.join(data)
uart.write(input)
pyb.delay(1000)
```

On the Raspberry Pi side, when a new data string is received, it will be splitted and saved in an list. If the len of this list is 3, which means all three values from the temperature, light and motion sensor are received, the current data list will then be appended to the database.

```
dataReceivedString = port.readline()
currentData = dataReceivedString.split(' ')
if len(currentData) == 3:
      data.append(currentData)
```

3.6 Taking photo

First we connected the Pi Camera to the Raspberry Pi. Then we had to enable the camera by enter at the command line

```
sudo raspi-config
```

In the menu select Enable Camera and reboot the devices. After that we installed the picamera library with apt:

```
sudo apt-get install python-picamera
```

After import picamera at the beginning of the python script, we create an instance of the PiCamera class. We set the default resolution at 1024x768. We used the Lux values received from the PiBoard to calculate the suitable iso of the camera. The iso value is set between 100 to 200 in normal light value, and 400 to 800 in low light environment.

```
if float(lux) >= 250:
	iso = 100 + (float(lux) - 250)/(1000 - 250)*(200-100)
else:
	iso = 400 - (float(lux) - 250)/(250)*(800-400)
	camera.iso = int(iso)
```

The current date and time will also be annotated in the image for debugging purpose so that we can distinguish between similar photos.

```
camera.annotate_text = strftime('%d-%m-%Y %H:%M:%S', localtime())
```

Finally we capture the photo and save as a file named image.jpg in the same project folder.

3.7 Editing photo

Our idea is to insert the current temperature on the photo, beside the icon of a thermometer. To do that, first we had to install and import the Python Imaging Library (PIL). We opened the original image as a base, and then paste the icon image on the base. We spent a lot of time adjusting the position of the icon and the temperature text on the image. We also changed the font family and font size of the text. Another interesting part is to add the degree sign ^{0} to the text. Finally we saved the output image to the current project folder and another copy to the web server folder in order to display the image on our website.

```
def editImage(temp):
	base_path = 'image.jpg'
	icon_path = 'icon.png'
	base = Image.open(base_path)
	icon = Image.open(icon_path)
	icon = icon.resize((200,200))
	offset = (0,550)
	base.paste(icon, offset, mask=icon)
	draw = ImageDraw.Draw(base)
	font_path= '/usr/share/fonts/truetype/freefont/FreeMonoBold.ttf'
	font = ImageFont.truetype(font_path,80)
	text = str(int(float(temp))) + u"\N{DEGREE SIGN}" + 'C'
	draw.text((180,620),text,(0,0,0), font=font)
	base.save(base_path)
	base.save('/var/www/html/image.jpg')
```

3.8 Building a website

Initially we tried to use Meteor, a framework which makes making JavaScript apps and website much faster and easier, especially when we already have some previous experiences building website with Meteor. We ended up spending a Tuesday trying to install it on the Raspberry Pi. If we could successfully installed it, we would have a cool images sharing website that we already built when taking a MOOC on Coursera, where users can sign up with their social network account, login, upvote or downvote all photos taken. It turned out Meteor did not support the processor of the Raspberry Pi, so that we just could not install it and had to give up.

We then moved on to setting up an Apache web server on the Raspberry Pi.

```
sudo apt-get install apache2 -y
```

A default web page is served at 10.94.66.236 - the IP address of our Raspberry Pi, and only desktop within the LAN can access the web page. We signed up at www.noip.com and get the hostname https://pibooth.ddns.net for our website. Now when the Raspberry Pi is connected to the internet, our web server will the up and running and every devices can access our web page at the new address pibooth.ddns.net.

Next we changed the default web page of Apache. We made it as simple as possible with a single background cover picture of a black photo booth curtain and the latest photo taken by the Pi Camera in the center. Later when we got the rrd data graph we insert a Bootstrap Carousel to display a slideshow of two pictures: the graph image and the Pi Camera photo. It will automatically change between these two images. Each time the graph is updated or a new photo taken, the web page will need to be refreshed in order to display new images.

3.9 Twitter API

We began with installing Twython on the Raspberry Pi, a Python module for interfacing with Twitter. Then we registered a Twitter application to get an API key. Finally we can use the Pi to tweet the image with the current temperature and light value as a status.

```
sudo pip install twython
```

Next, we created a Twitter account, then go to apps.twitter.com and create a new app. We gave our application access to our Twitter account, and were given an Access key and access secret.

```
CONSUMER_KEY = 'XPMRRREmy4jv6HU1U0N2d6HYv'
CONSUMER_SECRET = 'MHSkXmP28PW6IEVZ3iCitOlegWLhsZwLN9IBCd16dkOwBacUwU'
ACCESS_KEY = '722345011652796416-HtuggTLRcgv7DE5rUXrcJftNKLE04tz'
ACCESS_SECRET = 'LTH7mcHZs7H1gTY6M5s70RASEdcbZie2JTRLKR6BVAbVY'
Twitter=Twython(CONSUMER_KEY,CONSUMER_SECRET,ACCESS_KEY,ACCESS_SECRET)
```

We can now concatenate a string toTweet contains the current temperature and lux value, open the edited image then simply tweet it:

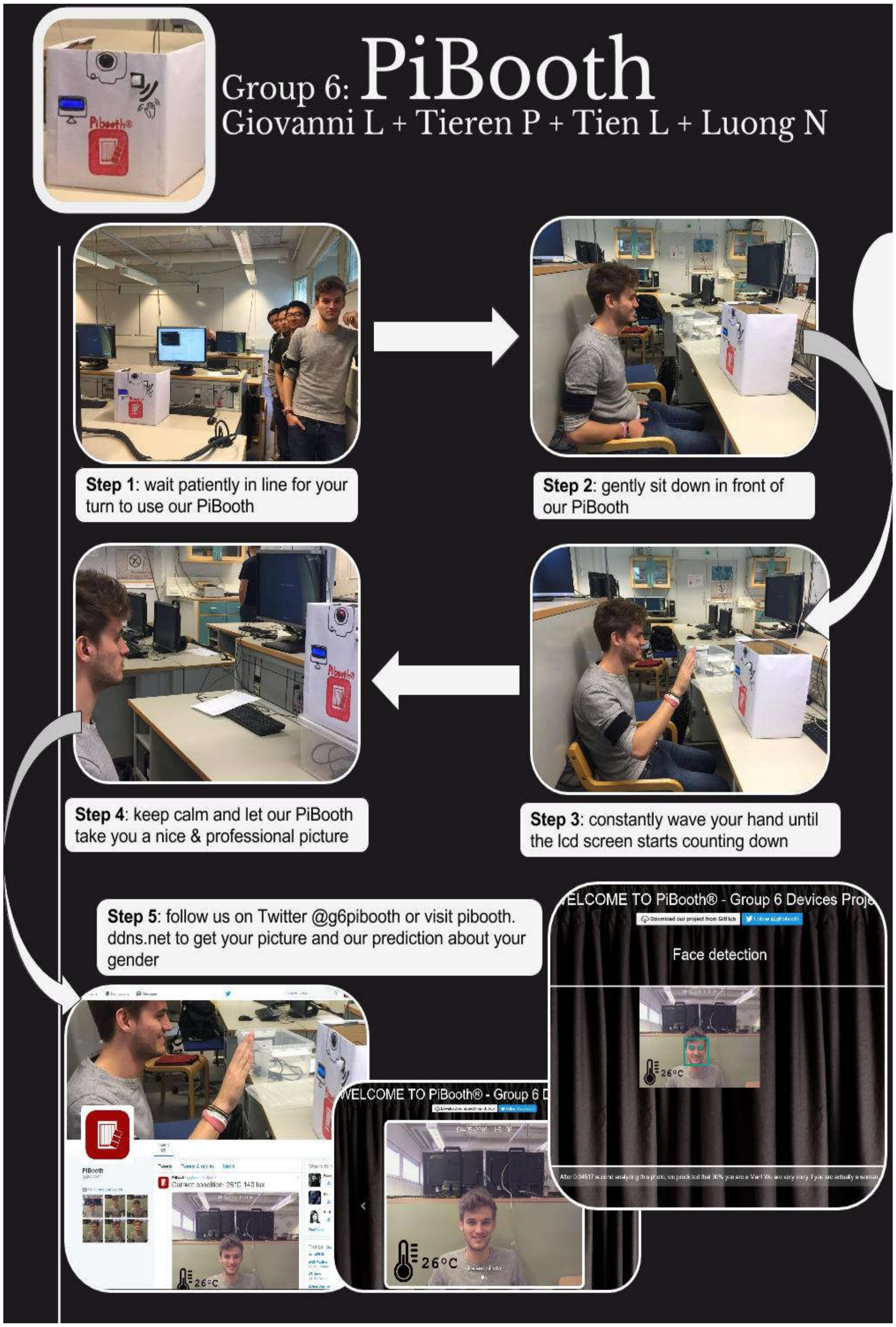
```
twitter.update_status_with_media(media=photo, status=toTweet)
```

After import the function tweet(temp, lux) in the main.py, after a photo is taken, it will be edited and automatically tweet it.

3.10 Face detection

For this feature initially we wanted to install OpenCV, a library aimed at real-time computer vision, but it would take 10 hours to compile it on the Raspberry Pi. We then searched for a cloud computing service that provide image analysis API and we ended up using Kairos. We registered an app, get the app id and app key, send the image and get a JSON response back. We just used the result of gender analytics, which returns the gender of the person in the photo with a certainty percentage. If no faces are recognized from the photo, we will print out “Ghost detected!”. This is our last feature and we hope to bring some fun to users, especially when we got wrong result.

4 User documentation


Group 6: PiBooth
Giovanni L + Tieren P + Tien L + Luong N

Step 1: wait patiently in line for your turn to use our PiBooth

Step 2: gently sit down in front of our PiBooth

Step 3: constantly wave your hand until the lcd screen starts counting down

Step 4: keep calm and let our PiBooth take you a nice & professional picture

Step 5: follow us on Twitter @g6pibooth or visit pibooth.ddns.net to get your picture and our prediction about your gender