Smart Home System using Rasperry Pi

Sebastian(Xin) HUANG
Shanghai Jiao Tong University
Student ID: 1162609010040

Xavier(Qiwei) XU
Shanghai Jiao Tong University
Student ID: 1162609010014

***Abstract*—In this report, we introduce a prototype of a smart home system using Rasperry Pi. This system is consisted of three parts: the fire disaster alarm system, intelligent night light system and outdoor rain prediction system. These systems are implemented with Rasbperry Pi as well as multiple sensors and actuators. What's more, we combined the data-mining technique with our system to enable it more intelligent. We can show in the demonstration that our implementation is promising in the tasks of fire alarming, controlling indoor light at night as well as rain prediction.**

I. Introduction

With the development of science and technology, we have stepped into the era of the Internet of Things(IoT). By linking objects together via Internet, the IoT has made our physical devices more intelligent and more controllable. Moreover, thanks to the rapid development of embedded system and information and communication technology, the Internet of Things becomes more and more commonly used in many aspects such as transportation system, medical management and intelligent house. For example, the invention of Apple watch has facilitated our lives in many aspects. Moreover, we have recently vitnessed the concept of the smart home, which describes a housing in which citizens are able to control electronic devices via their smart phone in a convenient way.

In this project, we design prototype of a smart home system based on Raspburry Pi. We make full use of sensors to measure multiple environmental parameters and based on these parameters we can turn on or turn off the electrical devices like light or alarming system automatically. Moreover, we combine the data-mining techniques into the prediction part of our system, making our system more intelligent.

In general, our prototype is composed of three parts: the intelligent night light system, the fire alarm system, and the outdoor rain forecast system. In our prototype, we would use two Raspberry Pis and a Arduino Uno 3 (as shown in the Figure I) as the control board with four kinds of sensors and two kinds of actuators.

II. Background

A. Internet of things

The Internet of things (IoT) is the inter-networking of physical devices, vehicles, buildings, and other items embedded with electronics, software, sensors, actuators, and network connectivity which enables these objects to collect and exchange data. Thanks to the development of advanced technologies

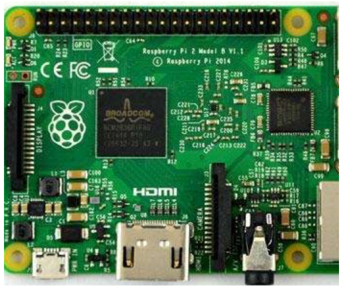
Fig. 1. Raspberry Pi

Fig. 2. Arduino Uno 3

including wireless communication, real-time analytics, machine learning, commodity sensors, and embedded systems, the Internet of Things has developed rapidly with a vareity of applications such as Environmental monitoring and Medical and healthcare devices[1]

B. Smart home

Smart Home is a building automation for home. It mainly involves the control of the lighting, heating, ventilation as well as the control of electronic appliances such as washer or oven. Moreover, it applies numbers of machine learning algorithms to do prediction and thus gives us hints which facilitate our lives.

C. Raspberry Pi

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation[2]. Combined with multiple sensors, the Raspberry Pi can realize a variety of applications in Internet of Things[3], including security cameras, intelligent indoor systems, etc.

III. Intelligent night light system

Controlling light automatically based on the surrounding situation is a small but useful task which brings much convenience to our lives. For example, we may have to move to the bathroom at night when the light is off. In this case, we may sometime struggling in finding the way to the bathroom in the dark. At this stage, we may ask why cannot the lighting system be more intelligent so that it can detect the arrival of people?

[1]https://en.wikipedia.org/wiki/Internet_of_things
[2]https://www.raspberrypi.org/
[3]https://www.raspberrypi.org/blog/tag/internet-of-things/

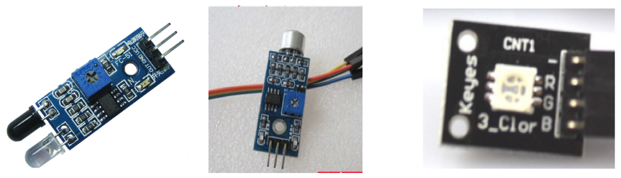

Fig. 3. left:Obstacle Sensor, middle:Sound Sensor, right:RGB LED

A. *Sensors and actuators*

In order to address this problem, we design this intelligent light system. Generally, this system is composed of two sensors and one actuators: a sound sensor, a light sensor as well as a RGB LED module. Since we do not have the PCF module which transforms the analog signal to digital signal, we would like to use the infrared obstacle detection module to represent the light sensor. The sensors or actuator that we use in this system are presented in the Figure 3.

B. *Function chart*

The function chart of the intelligent night light system is presented in the Figure 4.

Normally, when the light sensor detects that the environmental light intensity is less than the threshold, it gives a low level to the Raspberry Pi. In this situation, the Raspberry Pi would notice the level signal from the sound sensor. If sound sensor detects that there is enough environmental sound, it gives a high level to the Raspberry Pi. Then, the Raspberry Pi executes the actuation of the RBG LED module.

The RGB LED module is the only actuator in this system. It accepts three channels of PWM signal to produce light of three colors with variable light intensity. The duty ratio of the PWM signal determines the light intensity of LED module. As result, the user could adjust the light intensity of the night light according to their preference.

C. *Implementation detail*

The implementation of this system follows the condition 5. The Obstacle Sensor will detect whether it is luminous in the surrounding environment and recheck the light intensity after ten milliseconds. If there is not any light, the sound sensor will detect the sound. If the sound is louder than a threshold, the light will be automatically turned on. The main code can be described by following:

```
if(digitalRead(lightPin)==0){
   while (1){
      delay(10); //judge every 10 ms if there is
           light, if not, continue the process
      if(digitalRead(lightPin)==0){
          if(digitalRead(soundPin)==1){ /judge if
              there is sound, if yes, actuate the
              LED module
              ledColorSet(0xff,0xff);
              delay(5000);
              ledColorSet(0x00,0x00);
              break;
              }
          }
      }
  }
```

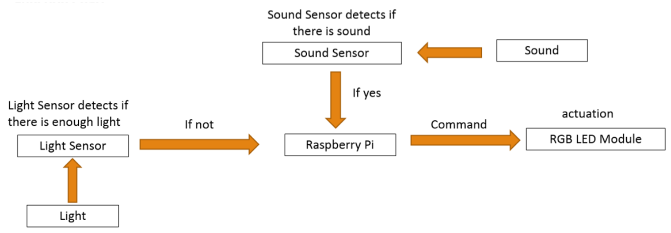

Fig. 4. The function chart of the intelligent night light system

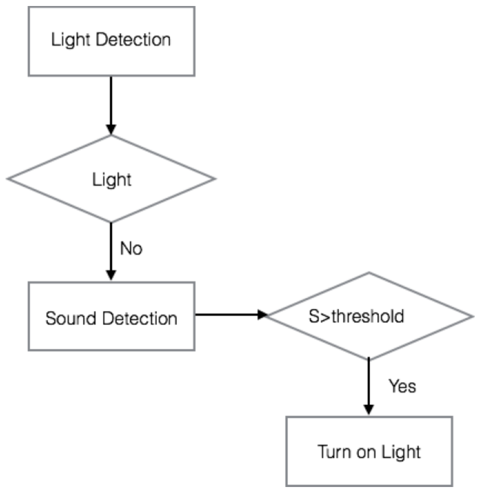

Fig. 5. Judging process

IV. Fire alarm system

Another problem that we should pay attention to in our everyday life is the precaution of the fire. Some severe fire begins with merely little smoke. Due to people's ignorance of the little smoke, the fire eventually turned to a fatal tragedy. Therefore, an alarm system to enhance peoples awareness of the fire at the very beginning is of great importance.

A. *Sensors and actuators*

In order to enhance the precaution of the fire disaster, we realize in this part the fire alarm system. In this system, we use one sensor and one actuator. The gas sensor is utilized to detect the smoke that is brought by fire. Then, a passive buzzer is invoked as the actuator. The gas sensor and the passive buzzer are presented in the figure 6 and the figure 7.

B. *Function chart*

The function chart of the fire alarm system is presented in the Figure 8,

The gas sensor detects real-time concentration of the environmental smoke. An analog signal is sent to the Arduino Uno3 Board to indicate the concentration of smoke. In this system, we make use of Arduino Uno3 Board since it is equipped with GPIOs which are able to convert the analog signal into digital signal. Then, when value of digital signal reaches the threshold, the control board sends a command to passive buzzer to play songs which warn us of the occurrence

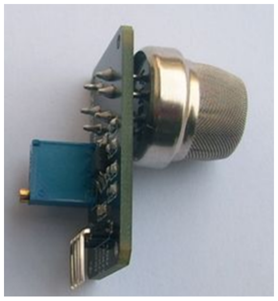

Fig. 6. Gas Sensor

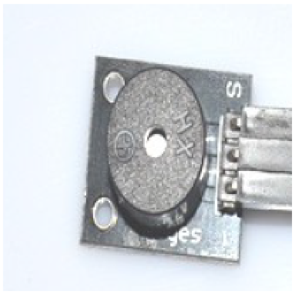

Fig. 7. Passive Buzzer

of the fire. Moreover, two thresholds have been set by the control board which indicates the slight smoke and severe smoke. The control board will command the passive buzzer to play different songs when the detected value overpasses different thresholds which realize a more accurate indication to people. The passive buzzer is the actuator of this system.

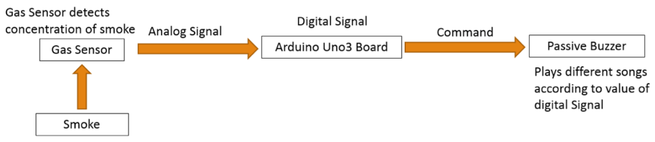
Gas Sensor detects concentration of smoke
Gas Sensor
Analog Signal
Digital Signal
Arduino Uno3 Board
Command
Passive Buzzer
Plays different songs according to value of digital Signal
Smoke

Fig. 8. The function chart of the fire alarm system

It is actuated by PWM signal to produce sound with different frequencies according to the PWM signal with different frequencies. That is why in our realization, the passive buzzer could play different songs according to the variation of input PWM signal.

C. Implementation detail

The implementation of this system follows the figure 9. It will conduct two *if* statements. After gas sensor collecs the data of smoke concentration, the control board will firstly check whether the concentration is above a higher threshold $thresholds_2$. If so, the passive buzzer will play song *Birthday Song*, otherwise, the second judgement will be evecuted. If the concerntration is above a lower threshold $thresholds_1$, The passive buzzer will play *Little Star*

V. Outdoor Rain Prediction System

Raining occurs frequently in our lives and in the raining days. We are suppose to take an umbrella to avoid being wet. For most people, the information about whether it will rain mainly comes from the weather forecast. However, the weather forecast generally gives a prediction on the weather of the city center, therefore, it may be less accurate for people living the rural area. Taking into account this situation, we design a localized rain prediction system.

The aim of the outdoor rain prediction system is to give users a hint about whether it will rain at the local area later. The principle of the prediction is based on the logistic regression. We firstly train a logistic regression model based on the online historical weather data, then we capture local weather condition via sensor. Given the local weather parameters, we are able to compute the probability of raining, and based on this probability, we give the hints to the users.

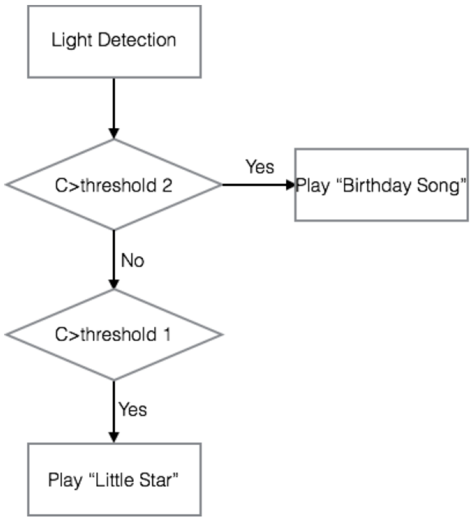

Fig. 9. Judging process

A. Infrastructure

The Infrastructure of the system is displayed in the figure 10. It is consisted of two Raspberry Pi with four main modules. Online data collection module, local data collection module, regression modules and the hint visualization modules. The local data collection module is built both on the client Raspberry pi and the server raspberry pi while the three other modules are implemented on the server Raspberry Pi.

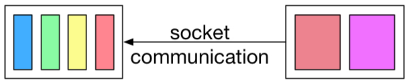

Fig. 10. Blue:Data Collection, Green:Regression module, Yellow:Data Visualization module, Red:Network communication, Purple:Local data collection

1) Online data collection module: The online data collection module aims at collecting real-time weather data from the official weather station and storing them in the local dataset. In this case, the local dataset can be updated periodically and thus improve the accuracy of the regression model. Thanks to Yahoo Weather API[4], we can fetch the real-time data via http request, we extract the information of temperature, humidity, pressure and whether it rains from the JSON file and appends it to the dataset.

2) Regression module: Given the dataset, this module aims at building a logistic regression to give the prediction of raining or not. Here we utilise three attributes of weather data, the temperature, the humidity and the pressure to build the

[4]http://developer.yahoo.com/weather/

model. In this way, we can represent the data sample with a 4-dimensional vector.

$$x = [temperature, humidity, pressure, 1]^T$$

Given parameters θ, the probability of raining can be represented as:

$$P(y = 1|x, \theta) = \sigma(\theta^T x) \tag{1}$$

Here, θ is the sigmoid function with

$$\sigma(x) = \frac{1}{1 - exp(-x)}$$

Similarly, we can describe the probability of no rain by using the following funciton.

$$P(y = 0|x, \theta) = 1 - \sigma(\theta^T x)$$

In order to fit this data, we need to estimate the parameters in the model, θ, the principle of parameters estimation is maximum log-likelihood. Thus, the equation 2 describes the function that we have to maximize.

$$L(x, y, \theta) = \sum_{i=1}^{m}(y^{(i)}log(h(x^{(i)})) + (1 - y^{(i)})log(1 - h(x^{(i)}))) \tag{2}$$

To train this model, we utilise the gradient descent algorithm. The main idea is that at each iteration, we update θ using the equation 3.

$$\theta_j := \theta_j + \alpha(y^{(i)} - h(x^{(i)}))x^{(i)} \tag{3}$$

To conclude, we adopt the algorithm 1 to train our model that gives the prediction. Given a new data x, we predict the

Algorithm 1 Gradiant Descent for Logistic Regression

```
Input:
  Dataset (X,y).
Output:
  The weight $\theta$.
  repeat
    for i in [0,3] do
      $\theta_j := \theta_j + \alpha(y^{(i)} - h(x^{(i)}))x^{(i)}$
    end for
  until converged or iteration limit exceeded;
```

probability using equation 1

3) Local Data collection module: This module contains two parts: data measure and data transmission. The data connection measure contains a sensor which is connected to the client. The temperature and humidity sensor measures the temperature and the humidity of the surroundings. After gathering the data, the client sends the data to the server via socket communication.

4) Data visualization module: This module visualizes the prediction result of our model. From the output of the regression model, we obtain the probability of rain. Therefore, by utilizing different colors of led light, we can express the sense of rain or no rain. In practice, we assign the red light to rain and the green light to no rain, as showed in Figure 11.

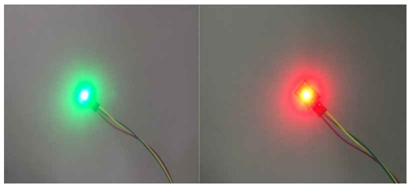

Fig. 11. Red light: Rain, Green light:No Rain

B. Workflow

The whole process can be described by the following flowchart 12.

- Firstly, the Online data collection module will connect the data from the internet every 30 minutes. After fetching the data, the new data will be appended to the file that stores the data in the format of temperature humidity pressure.
- Regression module is responsible for the training process, it runs once a day, and the prediction between the training is made based on the latest model.
- Every second, the sensors in Local Data collection module measure the weather parameters in the surroundings and these parameters will be sent to the server.
- The prediction will be made based on the data that the server receives. Meanwhile, the Data visualization module will scan the current state which indicates whether it will rain or not. Based on the state, it will turn red or green.

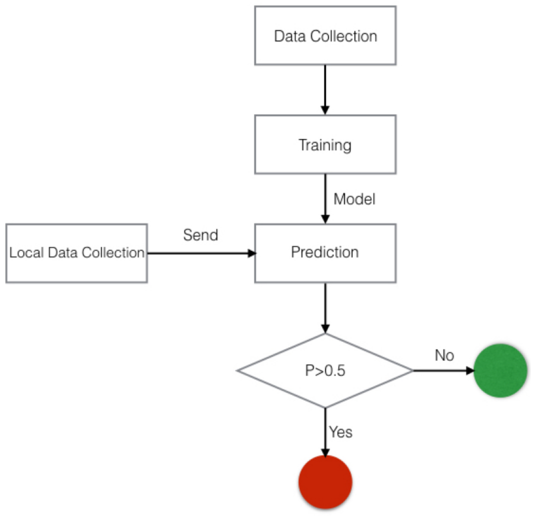

Fig. 12. Flowchart

C. Program Framework

Generally speaking, the program is made up by four threads: a main thread and three sub-threads.

1) main thread: The main thread plays a crucial role in the whole system. It can be comparable to the brain in the human body. It has two main functions. On one hand, it plays the role of a server in a network communication, listening and reading the data sent by the client. On the other hand, it is responsible for scheduling other three sub-threads. To be concrete, the main thread mainly contains two parts. For the part of network communication, the key code is displayed in the following column.

```
bind(listenfd ,(struct sockaddr*) &serv_addr , sizeof(
     serv_addr));
connfd = accept(listenfd , (struct sockaddr*)NULL,
     NULL);
int n=read(connfd ,sendBuff , sizeof(sendBuff));
```

Once a data is received, we call funciton *predict* to calculate the probability of raining and update the variable *state*.

```
state = predict();
```

For scheduling the sub-threads we utilise the method

```
pthread_create ( &thread[2], NULL, data_gathering ,
     NULL ) );
```

2) data collection thread: The implementation of data collection is made up of two parts. As the YAHOO support the API in python.therefore, the part charging for collecting data is written in python.

```
baseurl = "https://query.yahooapis.com/v1/public/yql
     ?"
yql_query_a = 'select item.condition , atmosphere
     from weather.forecast where woeid = 2151849'
yql_url_a = baseurl + urllib.urlencode({'q':
     yql_query_a}) + "&format=json"
result_a = urllib2.urlopen(yql_url_a).read()
data_a = json.loads(result_a)
```

Through the *http* query, we can obtain the current weather state of a specified city. Here, *2151849* is the world id of *Shanghai*. The data returned is stored in a dictionary data structure, therefore, we can get the attributes we need from the dictionary and append them into dataset file.

In order to call the python script, we write the function _System() to execute the python code and extract the result from the console log.

```
int _System(const char * cmd, char *pRetMsg, int
     msg_len)
```

Thus, the data gathering thread can be concluded as below.

```
void* data_gathering(void)
{
while(1){
   if(semaphore ==1){
     char *cmd = "python whe.py";
     char a8Result[128] = {0};
     int rett = 0;
     rett  = _System(cmd, a8Result , sizeof(a8Result))
        ;
     if(a8Result[0]-'0'==1){
         datalen++;
     }
   sleep(300);// successful update and sleep 300s
   }
     sleep(3);
   }
   pthread_exit ( NULL );
}
```

3) Training thread: The aim of this thread is to train the model. The thread is mainly consisted of three main parts.

- The *ReadData* funciton is for reading data from dataset.

```
ReadData(datalen ,  data , "./x2.txt","./y2.txt");
```

- The *Logistic* function is for training the dataset, after normalizing the data

```
Logistic( data ,logisW ,datalen);// function that
     trains the model
Normalize( data ,sampNum ); // normalize the
     dataset
logisW[j]-= alph*compute_gradiant(j , data ,logisW
     ); //compute the gradiant and update
```

4) Display thread: The *display* thread is for data visualization. It scans the variable *state* every ten seconds and based on the state, it sets the related color of LED light.

```
void *display(){
  while(1){
  pthread_mutex_lock ( &mut );
  if (state==1){
     ledColorSet(0xff ,0x00); //red
       }
     else if (state==0){
      ledColorSet(0x00 ,0xff); //green
       }
     pthread_mutex_unlock ( &mut );
     sleep(10);
  }
}
```

VI. Conclusion

In the project, we have proposed prototype for an intelligent house system which includes fire alarm system, intelligent night light system and outdoor rain prediction system. We implemented our design using two Raspberry Pi and one Arduino Uno3. The intelligent night light system ensures that the light turns on automatically only when the environment is dark and there is sound around; the fire disaster alarm system gives an accurate alarm when there is smokes. The outdoor rain prediction system has a satisfying predictive precision. In general, our system has a good performance in each circumstances.

VII. Future works

Our design is just a beginning with a wide range of future direction to excavate.

Firstly, in order to enhance the adaptability of the rain forecast system, along the path we travel, we could set several temperature and humidity sensors to gather weather data. As result, the rain situation along the path could be predicted by applying our rain prediction model. This would provide us rain prediction in a broader area instead of only the rain prediction in our living area.

Moreover, we would improve our prediction model by applying a more complex and accurate learning model.Some supplementary parameters which would be relevant to rain situation would also be considered such as the season. This makes our system more applicable.