Server

Antonio Mena, Logan Pasternak, Prayag Patel, Shreyas Ramachandran

Outline

- Project Objective
- Methodology
- Explanation of code
- Performance Evaluation
- Project Demonstration

Project Objective

- The goal of this project is to enable user access to real-time ambient data through a user-friendly interface
- The system will collect data on light, temperature, ar humidity using an RFID sensor system
- This data will be displayed on a web interface for users to access
- Send email notifications to user if sensor values surpass thresholds

Sensor

Light Sensor: Measures light intensity using diodes and the photoelectric effect

Temperature & Humidity Sensor: Measures capacitance, resistance, and thermal conductivity of the air to determine temperature and humidity

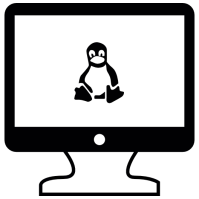
Radio Chip: Broadcasts data over radio waves

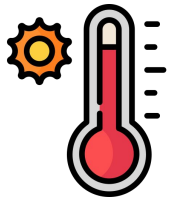
Methodology

- The Pip Tag made by Inpoint Systems has a light sensor and temperature/humidity sensor.
- The PipTag consists of a receiver and a transmitter:
  - The receiver requests the transmitter to send a signal with all the sensor values
  - The computer reads the response sent to the transmitter and parses it
- https://github.com/mingmingP/PIPtagCode
- A Linux operating system is required because of libusb
  - Libusb allows the computer to talk with the receiver

Approach (Setup)

- The codebase runs on a Linux virtual machine
- Using javascript we execute a command
- The receiver detects sensor readings from the transceiver using a C script
- Readings are sent to a localhost using socket.io to update the information as it comes to the receiver
- If the values of any sensors pass a predefined threshold, an email is sent to a user using nodemailer





Approach (VirtualBox)

1

Create Virtual Machine

Ours was done on Virtual Box and a clean install on a laptop

2

Configure USB

Add USB drivers to allow the machine to read from the Receiver

3

Download Repository

We got a majority of the code from Professor Chen

4

Download Libraries

C Libraries: g++, libusb

Js Libraries: nodemailer, http, socket.io, express

5

Compile code

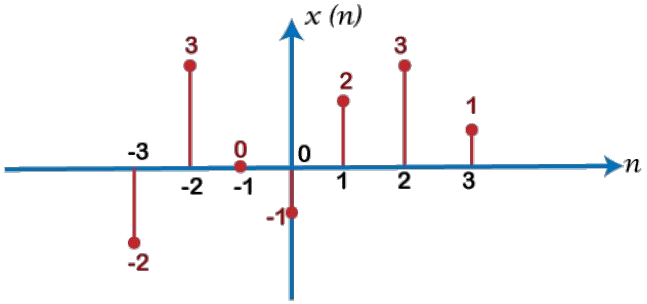
Compile ReceiverCode using g++

6

Run on Node.js

Run server script using Node.js and view data on localhost

Approach (Data Handling)



- Data parsed from bash script
- Measurement thresholds set for humidity, temperature, and light measurements
- Signal Processing
  - Server initializes buffer of size five for each measurement
  - Server serves the mean filtered results from measurements
- Measurement Thresholding
  - Three counters initialized which represent the number of times a certain measurement threshold was surpassed.
  - Once counters surpass a certain threshold, an email is sent to the client’s email address

Approach(Email Handling)

- The sending of emails was facilitated through the use of the nodemailer package.
- Configuring gmail account to send emails
  - Create a gmail account
  - Set up two-factor authentication for your account
  - Create an app password
- Create an environment file in your local directory (.env)
  - SENDER_EMAIL, RECIPIENT_EMAIL, APP_PASSWORD
- Create and run nodemailer transport protocol

Explanation of Code

Executes command and defines thresholds

```
60
61  // Sensor data processing
62  const cmd = 'sudo stdbuf -o0 ./ReceiverCode/pip_sense.v2 1 1 | stdbuf -o0 grep TX:0$1 | tee $2';
63  const out = exec(cmd);
64  let times = 0;
65  let oTimes = 0;
66  let flag = 0;
67
68  let rssiBuffer = [];
69  let lightBuffer = [];
70  let tempBuffer = [];
71  let humidityBuffer = [];
72
73
74  const flag_threshold = 5;
75  const buffer_size = 10;
76
77  let light_counter = 0;
78  let temp_counter = 0;
79  let humidity_counter = 0;
80
81  const temperature_threshold_high = 30;
82  const temperature_threshold_low = 15;
83  const humidity_threshold_high = 70;
84  const humidity_threshold_low = 20;
85  const light_threshold_high = 1000;
86  const light_threshold_low = 50;
87
```

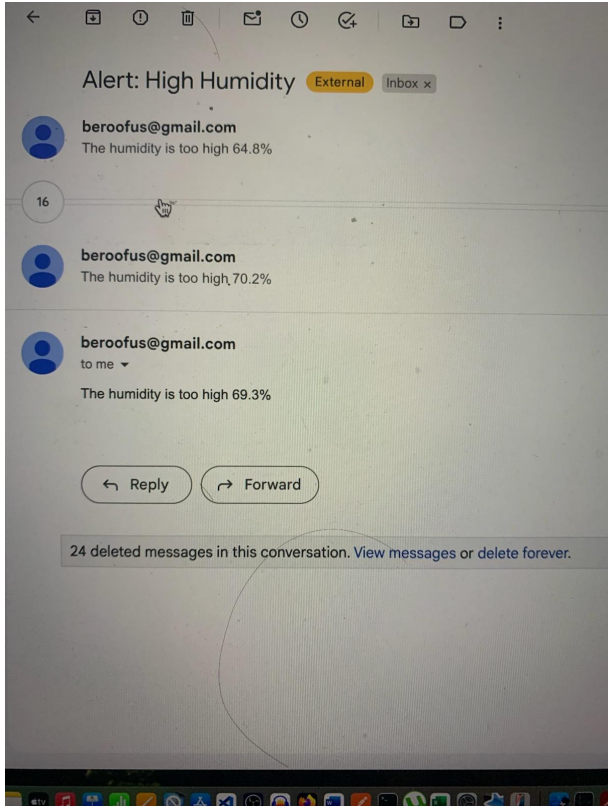
Extracts specific data values from the command output and converts them to numbers.

```
88  function calcMean(buffer){
89          return buffer.reduce((sum,value) => sum + value, 0) / buffer.length;
90  }
91
92  out.stdout.on('data', (line) => {
93    const rssi = Number(line.substring(line.lastIndexOf('RSSI') + 5, line.lastIndexOf('C') - 7)) || 0;
94    const lumens = Number(line.substring(line.lastIndexOf('light') + 7, line.lastIndexOf('temp') - 1)) || 0;
95    let temp = Number(line.substring(line.lastIndexOf('temp') + 6, line.lastIndexOf('humidity') - 1)) || 0;
96    let humid = Number(line.substring(line.lastIndexOf('humidity')+10)) || 0;
97
98
99    //Calibrate and adjust temp and humidity value readings
100   const tempValue = temp - 6.4;
101   const humidity = humid/10;
102
103   console.log(`RSSI VALUE: ${rssi}`);
104   console.log(`LIGHT SENSOR: ${lumens}`);
105   console.log(`TEMPERATURE SENSOR: ${tempValue}`);
106   console.log(`HUMIDITY SENSOR: ${humidity}`);
107
108   io.emit('sensorData', { rssi, lumens, temp: tempValue, humidity });
```

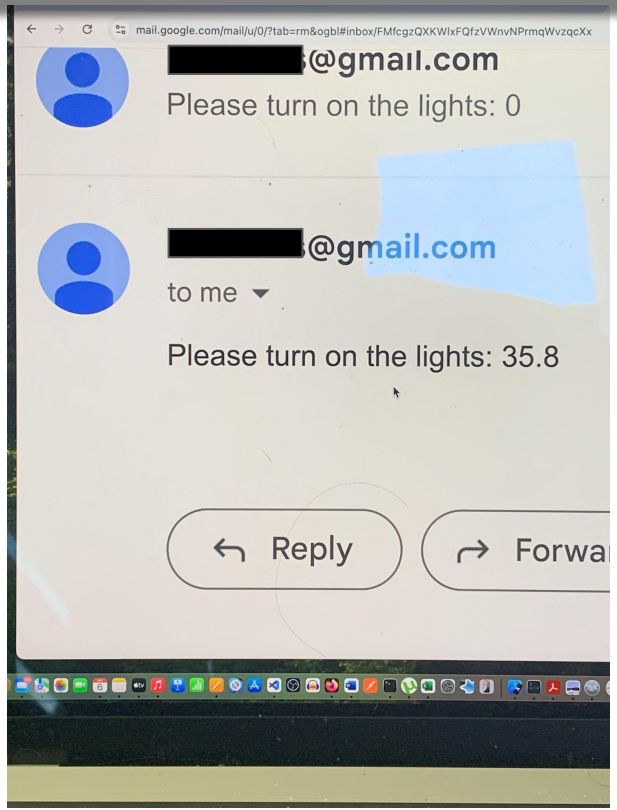
Performance Evaluation

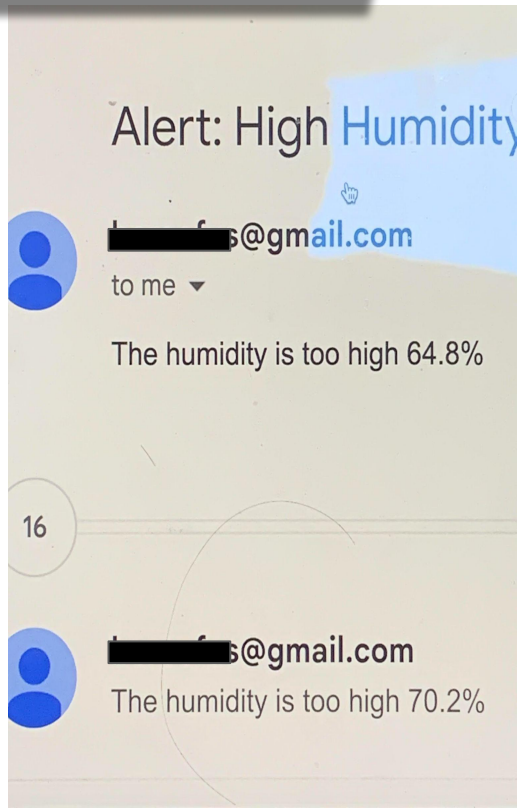
- The RFID Server is working properly, providing an accurate and real-time data collection and updates
- Most of our changes involved fixing issues with C scripts and adding JavaScripts(JS) scripts to parse the command output and calibrate the data

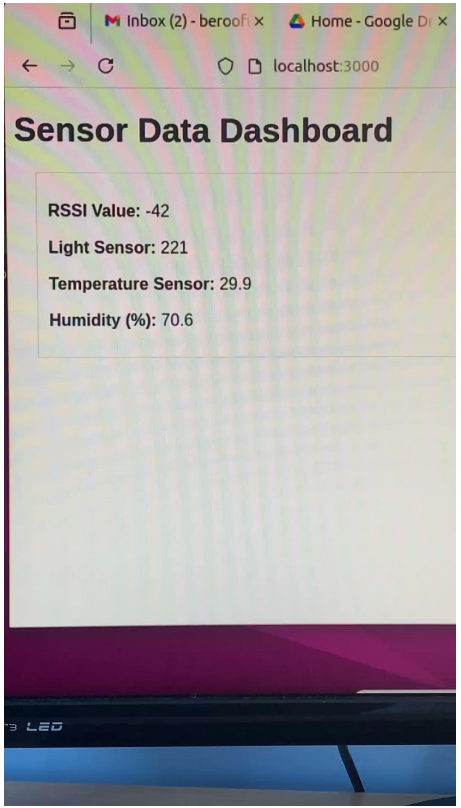
Project Demonstration



Project Demonstration







Conclusion

In summary, our team's RFID server is able serve mean-filtered temperature, humidity, and lumen data from our Pip Tag to a local host. Additional, it can send email notifications to clients given measurement thresholds are surpassed a given number of times. RFIDs, although simple, are powerful tools that allows easy communication between sensors and computers

Future Work:

- Data collection set up with an S2 Bucket
- Incorporate classification model to add features to email services
    - Predicted temperature, type of clothing, etc

Future Work - Classification

Our classification algorithm depends on the task:
Recommend clothing - Decision Tree

Classify patterns such as light signals(Morse Code) - Matrix Profile or DTW

Forecast weather - Regression methods

Contributions

Logan Pasternak: Email Notification Setup, Measurement Filtering

Antonio Mena: Setting up the pipsense code and verifying how it works.

Prayag Patel: Identified reasons why Linux was necessary, changed out battery, worked on JS script to send data to localhost

Shreyas Ramachandran: RFID pip tag/receiver hardware functioning, testing

Q&A

Why do the values fluctuate so much?

The values are added together and averaged. Most likely, the values fluctuate because of our code. When the values are added and averaged in real-time, there can be fluctuations.

Note: Although sensors stray out of calibration overtime, the sampling rate does not change.

```
88    function calcMean(buffer){
89            return buffer.reduce((sum,value) => sum + value, 0) / buffer.length;
90    }
```

```
109
110   //Add data to buffers
111   if (lightBuffer.length >= buffer_size){
112           lightBuffer.shift();
113   }
114   if (tempBuffer.length >= buffer_size){
115           tempBuffer.shift();
116   }
117   if (humidityBuffer.length >= buffer_size){
118           humidityBuffer.shift();
119   }
120
121   lightBuffer.push(lumens);
122   tempBuffer.push(tempValue);
123   humidityBuffer.push(humidity);
124
125   //Calculate means
126   const avgLumens = calcMean(lightBuffer);
127   const avgTemp = calcMean(tempBuffer);
128   const avgHumidity = calcMean(humidityBuffer);
129   console.log(`Avg Lumens: ${avgLumens}, AvgTemp: ${avgTemp}, Avg Humidity: ${avgHumidity}`);
```