Team Guugle

Milestone 4

Members: Andrea Chamorro, Kai Johnson, Lucas Laughlin, Yushen Chang, Matt Cohen

The project is to make robot that travels around and gathers temperature data. The original project involved using an Arduino Uno R3 to store the data over bluetooth into an excel file through a serial terminal. However, the Arduino Uno R3 lacks Wi-Fi capability which makes database collection much more complicated. We decided to use the Raspberry Pi 3 B+ as a microcontroller instead. The Raspberry Pi 3 B+ allows for Wi-Fi connectivity, which allows for a database to be stored directly on it. The Raspberry Pi 3 B+ also allows us to upload the contents of the database directly on a server such as Apache or Heroku.

Functionality

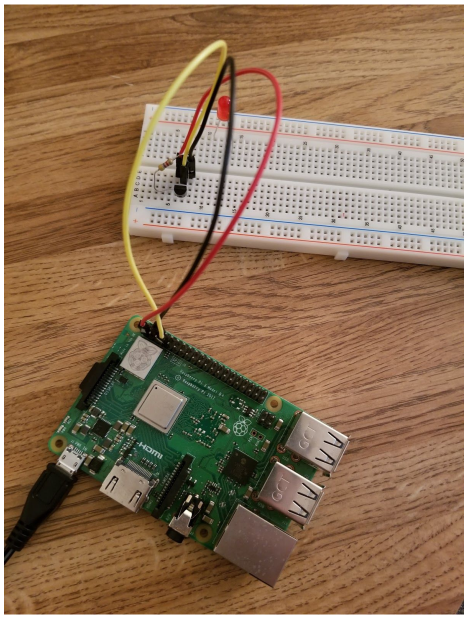

Figure 1: Raspberry Pi 3 B+ reading temperature data from DS18B20 temperature sensor

The Raspberry Pi 3 B+ uses an operating system known as Raspian which is Debian based. This microcontroller is highly compatible with most programming languages, because it’s operating system is Linux based.

For milestone 4, we wrote a Python script called “databasev2.py” which reads temperature data from the DS18B20 temperature sensor and logs the data into a database (sensorlog.db). The database management system of choice is SQLite3 because it is highly compatible and lightweight compared to Postgresql.

Database Script

Link to the scripts: https://github.com/Natty-Laugh/final-project/tree/master/milestone4
The database script “databasev2.py” is programmed in Python to interface with the hardware of the Raspberry Pi and the SQL database. The script works by taking the temperature data from the DS18B20 and storing it into an SQL table for storage. The Python script stores the temperature data every time the script is executed. The following is the output of the data stored in the table temp after running the python script:

```
pi@raspberrypi:~/milestone4 $ python databasev2.py
26.937
temperature = 26.937
temperature = 26.937
2018-11-19 02:58:26.937
2018-11-19 03:27:26.937
2018-11-19 03:30:26.937
2018-11-19 03:32:25.342
```

The code above show the contents of table temp of the sql database. The format of the data that is logged into the SQL database is the date, time, and temperature. The table is created as follows:

```
pi@raspberrypi:~/milestone4 $ sqlite3 sensorlog.db
sqlite > BEGIN;
sqlite > CREATE TABLE temps (timestamp DATETIME, temp NUMERIC);
sqlite > COMMIT;
```

This code creates a sql table which allows for the Python script to store time and temperature data into the database.

The next aspect of the script is how Python reads temperature data into the sql database. The DS18B20 is connected to the Raspberry Pi through GPIO pins. GPIO pins are hardware pins used for programmable input/output functionality. The DS18B20 consists of 3 pins a power pin, ground pin, and a data pin. All 3 pins interface with the Raspberry Pi through GPIO. The data pin of the DS18B20 is what provides the data and is needed for the Python script. In order for the Python script to read data it needs to know the address location of the DS18B20. In the Python script we must include the device file path /sys/bus/w1/devices. This is the path for the script to get temperature data. The following snippet of code searches for the device path:

```
devicelist = glob.glob('/sys/bus/w1/devices/28*')
```

The next component is using Python to store the data into sql, this is done through using the cursor class in Python. The cursor class gives Python the capability to use sql commands. The following lines of code allow for the script to do exactly that:

```
#connect class is used to connect to sql database also used for socket programming
db_connect = sqlite3.connect(myDatabase)
#cursor class allows Python to execute PostgreSQL command in database session
db_cursor = db_connect.cursor()
db_cursor.execute("INSERT INTO temps values(datetime('now'), (?))", (temp,))
```

To summarize, Python connects to the database and uses the cursor class to give Python a way to use sql commands and then store the data in the database.

Database Model

Our database consists of temperature vs. time data which will be displayed in graphs in our front end and which we collected from the Raspberry Pi. Additionally, we have a login option in our front end, which collects username and password data, which is stored once per user. Every time a user collects and graphs data it will get stored in our sessions table, and they will be able to add notes about the individual session, as seen on the model . Each session maps to one user, and a user can have many sessions. There is one-to-one mapping between sessions and temperature tables. This model was created using MySQL Workbench.

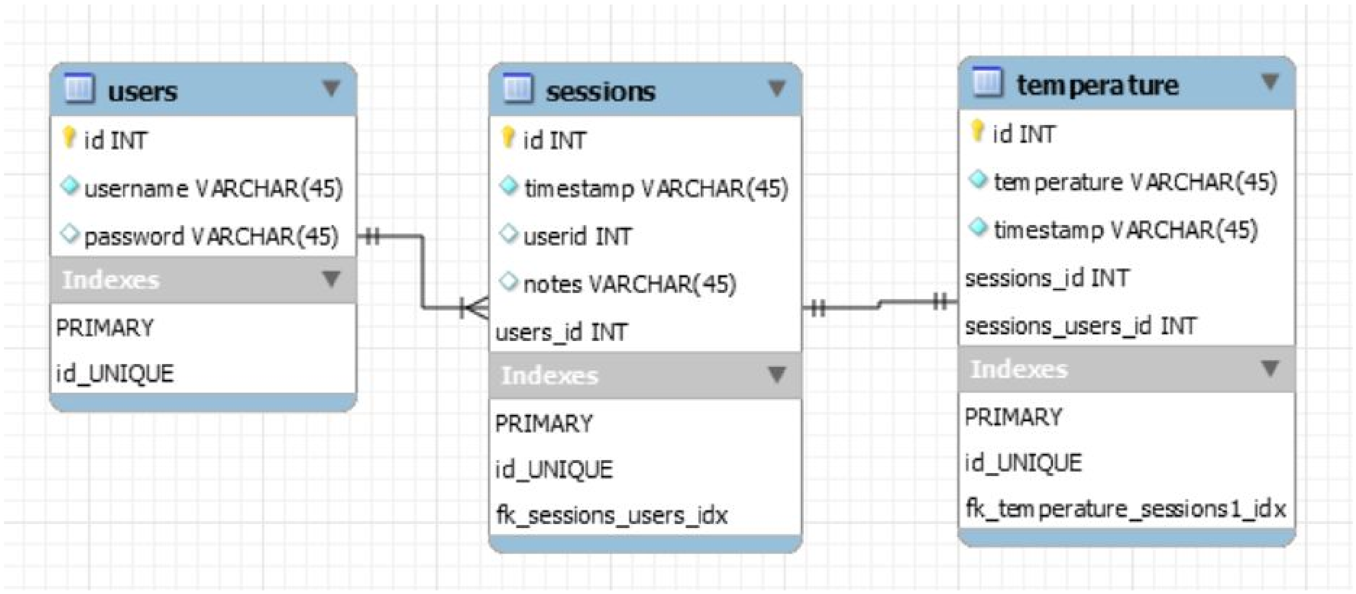

Figure 2: Relationship of database table.

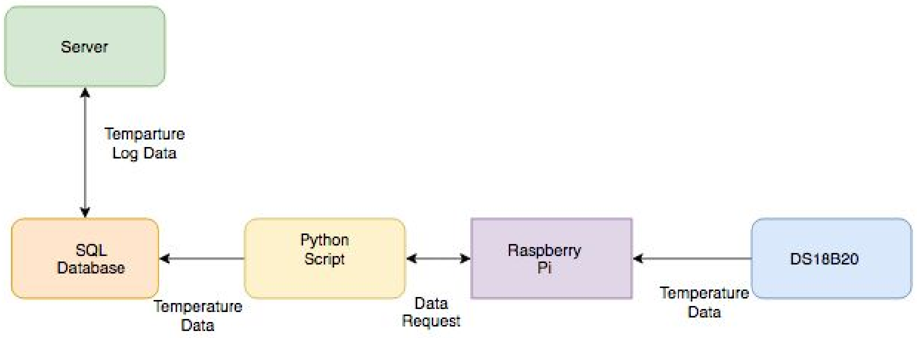

Figure 3: Simple relational block diagram of how each module and the signals that transfer between them.