Experiment No. 03:

Wireless Data Acquisition Using Temperature Sensor

By: Abhilash Kashyap Balasubramanyam

Lab Project Partner: Adnan Patel

Instructor: Dr. Jafar Saniie

ECE 510

Lab Date: 05-30-2024

Due Date: 06-07-2024

Acknowledgment: I acknowledge all of the work (including figures and codes) belongs to me and/or persons who are referenced.

Signature: [illegible]

I. Introduction

A. Purpose

The basic method for using two distinct embedded system platforms to exchange data via Bluetooth protocol is introduced in this lab. It is based on either a digital external sensor that interfaces through digital input/output pins or an analogue external sensor that interfaces through an on-board analog-to-digital (ADC) converter.

Building an ambient temperature detection system utilising an external sensor that interfaces with an Arduino board is necessary for this lab. In order to show incoming temperature sensor data, a central node for wireless sensor data collection must also be set up using a Raspberry Pi single-board computer. This node will communicate with the Arduino board over the Bluetooth protocol. To enable remote access to the acquired data, the experiment must additionally link the centrally collected data to ThingSpeak, a cloud computing server.

B. Background

Raspberry Pi

The Raspberry Pi Foundation is a British company that developed the Raspberry Pi line of small single-board computers. Promoting the teaching of fundamental computer science in classrooms and in underdeveloped nations was the goal. The initial model sold outside of its designated market for applications like robotics and became significantly more popular than expected. Cases and peripherals (such keyboards and mouse) are not included. However, several official and unofficial packages have included certain accessories. A System on a Chip (SoC), or a whole computer on a single chip, is what powers the Raspberry Pi. It includes a CPU, graphics processing unit (GPU), memory, and all other required parts. The Broadcom BCM2835 SoC, which powers the Raspberry Pi, has an ARM1176JZF-S 700 MHz CPU, a Video Core IV GPU, and 512 MB of RAM.

The General-Purpose Input/Output (GPIO) pins of the Raspberry Pi are one of its most important features. These serve as the Pi's physical interface with the outside world. They can be thought of as switches, either input or output, at their most basic level. Beyond that, though, GPIO pins can communicate with a vast array of parts, from basic LEDs and buttons to intricate sensors, displays, and even the internet.

Forty GPIO pins of the Raspberry Pi are used to connect motors, lights, sensors, and other devices. Compared to the original Raspberry Pi's 26-pin GPIO header, this is a major gain. Every pin is unique and has multiple applications, including as GPIO, I2C, SPI, UART, and more. On top of the Raspberry Pi, close to the yellow video out socket, are the GPIO pins. There are two rows of twenty pins each, organized in a 2x20 layout. This facilitates the connection of a large variety of accessories and gadgets to the Raspberry Pi.

In addition, the Raspberry Pi features several connectors that let it establish connections with other gadgets and the internet. These consist of an SD card slot, an Ethernet port, an HDMI port, and a USB port. While the HDMI port can be used to connect a display, the USB port is used to attach accessories like a keyboard and mouse. You can load the operating system and save data on the SD card slot in addition to connecting to the internet via the Ethernet connector. To sum up, the Raspberry Pi is an incredibly strong and adaptable gadget that has a plethora of uses. It is an excellent tool for learning computer science and electronics because of its GPIO pins and ports, which enable it to communicate with a variety of devices and the internet.

Arduino UNO

Arduino is an open-source company that designs and manufactures microcontrollers and kits for creating digital devices and interactive objects. These boards can be connected to expansion boards and other circuits using their digital and analog input/output pins. Additionally, they provide serial communication interfaces for program loading, like USB. The most common programming languages for microcontrollers are C and C++. Furthermore, an integrated development environment based on the Processing Language Project is made available through the Arduino project.

HC-06 Bluetooth Module

A popular wireless communication tool that allows serial communication between a microcontroller and a Bluetooth-capable device, like a computer or smartphone, is the HC-06 Bluetooth module. The HC-06 is a Bluetooth 2.0 device that is primarily intended for short-range wireless data sharing. It provides dependable and simple serial connectivity.

The module can only reply to requests from a master device because it is a slave-only device, which means it cannot start a conversation on its own. It uses UART (Universal Asynchronous Receiver/Transmitter) to interface with microcontrollers; the default baud rate is 9600 bps, although this can be changed. The HC-06 is easy to integrate into a variety of electronic applications due to its straightforward 4-pin interface, which consists of VCC, GND, TX (Transmit), and RX (Receive) connections.

The HC-06's simplicity of use is one of its main benefits. With the use of conventional AT commands, users can change parameters like the passkey, baud rate, and device name. Many microcontrollers, including Arduino and Raspberry Pi, are compatible with the module because it normally runs at a voltage between 3.3V and 5V.

The HC-06 is perfect for applications like home automation, wireless sensors, and robotics that need wireless control or data logging because of its roughly 10-meter range. The module's low power consumption and small size further improve its applicability for battery-powered and portable devices. In general, the HC-06 Bluetooth module provides an affordable and dependable option for wireless serial communication across a range of electronic uses.

DHT 22 Temperature Sensor

Digital sensors like the DHT22, or AM2302, are widely used to measure humidity and temperature. It is a preferred option for many environmental monitoring and control applications because of its great dependability and long-term stability.

The DHT22 sensor measures the air quality around it using a thermistor and a capacitive humidity sensor. Its single-wire protocol allows it to output calibrated digital signals, which facilitates its interface with microcontrollers like the Arduino, Raspberry Pi, and other development boards. The sensor uses 1.5 mA of power when measuring and 0.1 mA when it is idle. It functions within a voltage range of 3.3 V to 5 V.

In terms of performance, the DHT22 offers a humidity range of 0 to 100% RH with an accuracy of ±2-5% RH and a temperature range of -40 to 80°C with ±0.5°C. Because of its bi-second data delivery design, it can be used in applications that need real-time environmental monitoring.

Four pins make up the sensor: a not-connected pin, GND, DATA, and VCC. The microcontroller receives the measured values via the DATA pin. The sensor is a single-wire device, so for it to work properly, there needs to be a pull-up resistor (usually 4.7kΩ) between the DATA pin and VCC.

To provide reliable and accurate measurements, the DHT22 is housed in a sturdy plastic shell that shields it from dust and other impurities. Because of this, it is perfect for usage in dehumidifiers, weather stations, HVAC systems, and other applications where humidity and temperature monitoring are critical. All things considered, the DHT22 provides an affordable, precise, and user-friendly way to measure environmental variables.

II. Lab Procedure and Equipment List

A. Equipment

Equipment

- 1 x Raspberry Pi Single Board Computer
- 1 x Arduino UNO
- 1 x DHT 22 Temperature/Humidity Sensor
- HC-06 Bluetooth Module
- Arduino IDE
- ThingSpeak Portal

B. Procedure

Temperature Sensing on Arduino Board

1. The connections using Arduino Uno along with HC-06b Bluetooth Module, DHT 22 Temperature sensor is set up.
2. On the Arduino IDE, the first program is executed where establishment of connection is complete.
3. Hereafter, using the 'AT', 'AT+NAME', 'AT+PIN' and 'AT+BAUD' commands are used to test the connection, set a name to the Bluetooth module, and setting a pin to the Bluetooth module.
4. At this point, the second program on the Arduino IDE is executed where, the temperature measured by the DHT 22 sensor is displayed both in Celsius and Fahrenheit which is transmitted to the Bluetooth module.

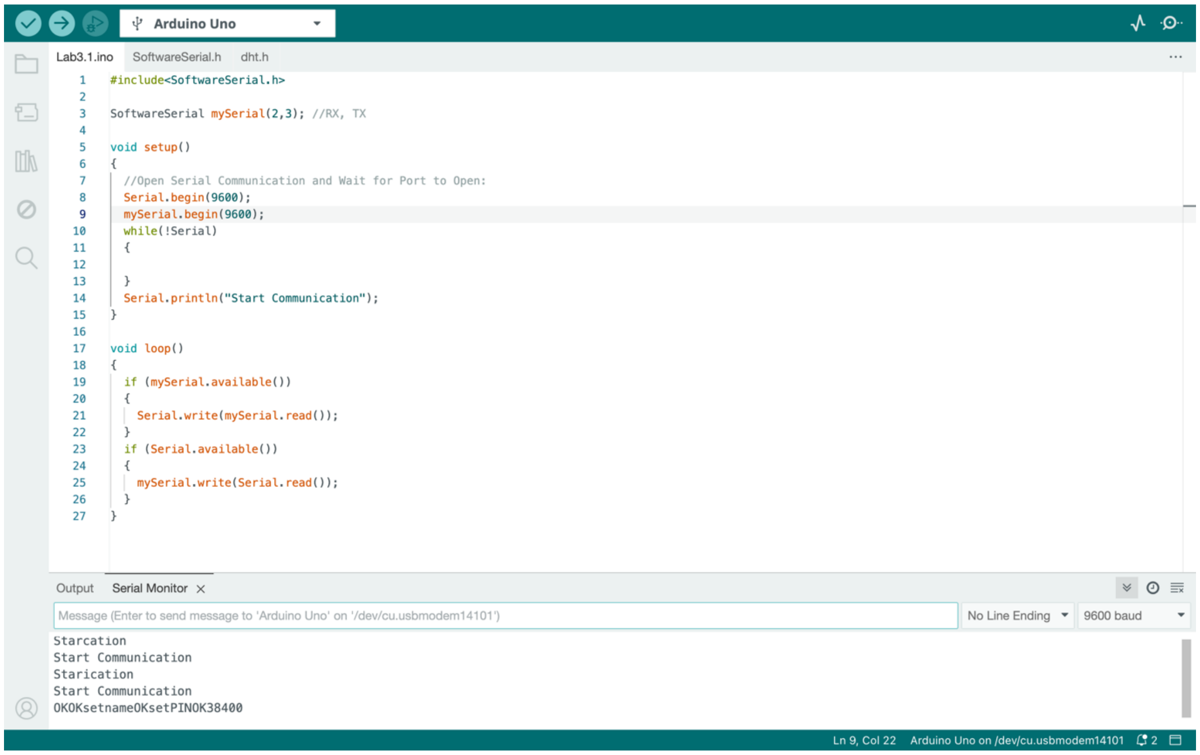
Connecting the RaspberryPi to the Arduino Board through the Bluetooth module

1. Upon connecting the Pi to the internet, the Bluetooth control is entered into using the ‘bluetoothctl’ in the terminal.
2. Upon initialization, scanning for nearby Bluetooth devices is begun and the name assigned to the Bluetooth module is identified and the MAC address of the same is noted.
3. Then, entering ‘pair <copied MAC address>’ the raspberry pi is paired with the Bluetooth module.
4. Upon exiting from the Bluetooth command screen using ‘exit’ command, the next set of commands are ‘sudo apt update’ -> ‘sudo apt install bluetooth libbluetooth-dev’ -> ‘sudo python3 -m pip install pybluez’ -> ‘sudo python3 -m pip install thingspeak’ for initializing the Bluetooth functions for python.

Connecting the system with Thingspeak

1. A user account is to be created on Thingspeak and a new channel is created.
2. Field 1 is set to Temperature and the API key along with the Channel ID is recorded and the respective fields are changed in the Python Program on the Raspberry Pi.
3. On the terminal, ‘python3 <program_name.py>’ is executed and the resultant temperature reading starts to display on the terminal on the Raspberry Pi.
4. The same temperature is also recorded on the ThingSpeak portal as well with a corresponding graph of the rise and fall in temperature is visualized.

III. Result Screenshots and Video Demonstration



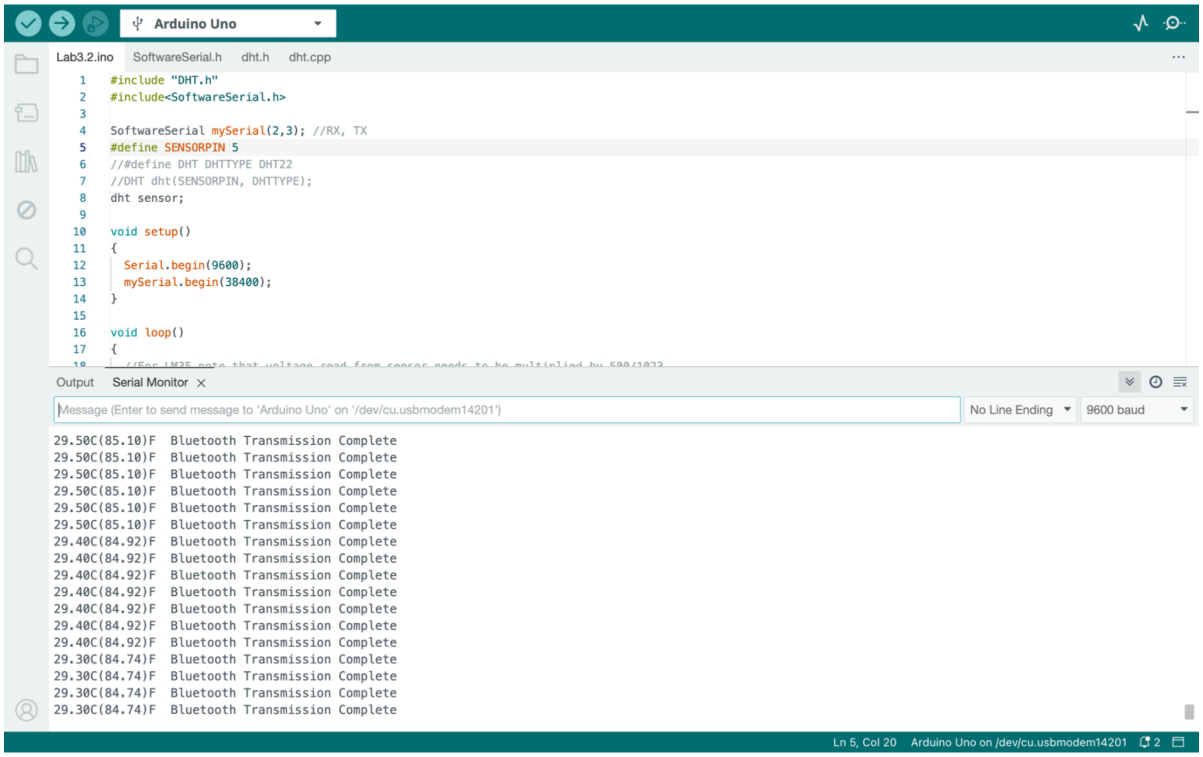
```
#include<SoftwareSerial.h>

SoftwareSerial mySerial(2,3); //RX, TX

void setup()
{
  //Open Serial Communication and Wait for Port to Open:
  Serial.begin(9600);
  mySerial.begin(9600);
  while(!Serial)
  {

  }
  Serial.println("Start Communication");
}

void loop()
{
  if (mySerial.available())
  {
    Serial.write(mySerial.read());
  }
  if (Serial.available())
  {
    mySerial.write(Serial.read());
  }
}
```

Message (Enter to send message to 'Arduino Uno' on '/dev/cu.usbmodem14101') — No Line Ending — 9600 baud

```
Starcation
Start Communication
Starication
Start Communication
OKOKsetnameOKsetPINOK38400
```

Ln 9, Col 22 Arduino Uno on /dev/cu.usbmodem14101

Figure 1: Initial connection setup with device name and PIN.



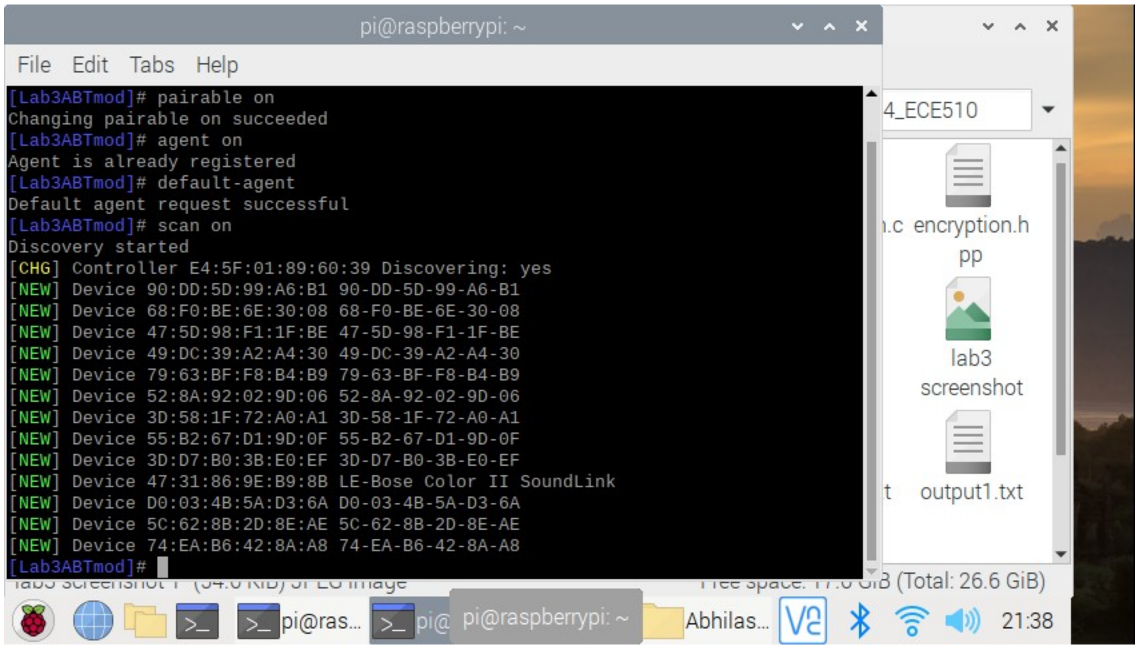
```
#include "DHT.h"
#include<SoftwareSerial.h>

SoftwareSerial mySerial(2,3); //RX, TX
#define SENSORPIN 5
//#define DHT DHTTYPE DHT22
//DHT dht(SENSORPIN, DHTTYPE);
dht sensor;

void setup()
{
  Serial.begin(9600);
  mySerial.begin(38400);
}

void loop()
{
  //For LM35 note that voltage read from sensor needs to be multiplied by 500/1023
```

Message (Enter to send message to 'Arduino Uno' on '/dev/cu.usbmodem14201') — No Line Ending — 9600 baud

```
29.50C(85.10)F  Bluetooth Transmission Complete
29.50C(85.10)F  Bluetooth Transmission Complete
29.50C(85.10)F  Bluetooth Transmission Complete
29.50C(85.10)F  Bluetooth Transmission Complete
29.50C(85.10)F  Bluetooth Transmission Complete
29.50C(85.10)F  Bluetooth Transmission Complete
29.40C(84.92)F  Bluetooth Transmission Complete
29.40C(84.92)F  Bluetooth Transmission Complete
29.40C(84.92)F  Bluetooth Transmission Complete
29.40C(84.92)F  Bluetooth Transmission Complete
29.40C(84.92)F  Bluetooth Transmission Complete
29.40C(84.92)F  Bluetooth Transmission Complete
29.40C(84.92)F  Bluetooth Transmission Complete
29.30C(84.74)F  Bluetooth Transmission Complete
29.30C(84.74)F  Bluetooth Transmission Complete
29.30C(84.74)F  Bluetooth Transmission Complete
29.30C(84.74)F  Bluetooth Transmission Complete
```

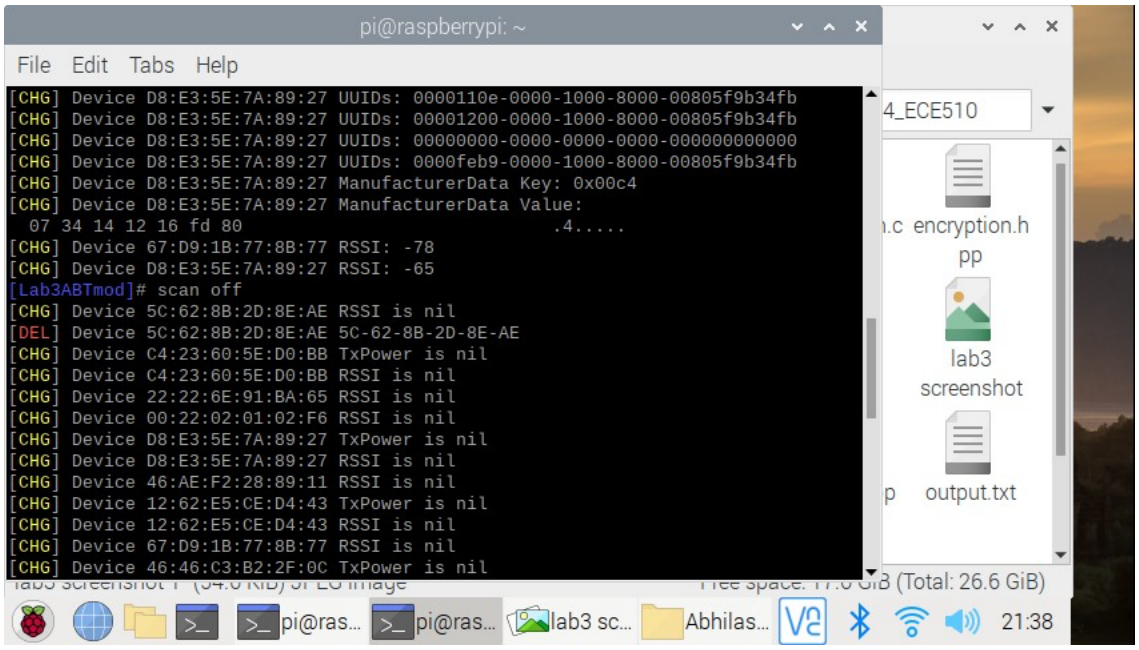
Ln 5, Col 20 Arduino Uno on /dev/cu.usbmodem14201

Figure 2: Recording of Temperature Readings.



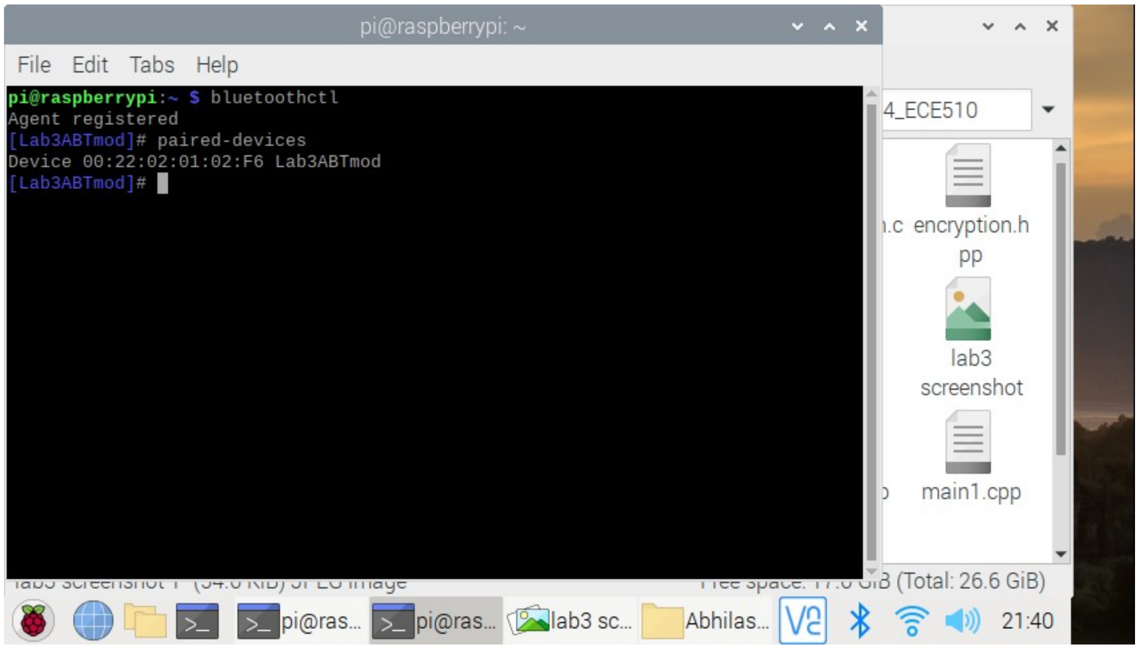
```
[Lab3ABTmod]# pairable on
Changing pairable on succeeded
[Lab3ABTmod]# agent on
Agent is already registered
[Lab3ABTmod]# default-agent
Default agent request successful
[Lab3ABTmod]# scan on
Discovery started
[CHG] Controller E4:5F:01:89:60:39 Discovering: yes
[NEW] Device 90:DD:5D:99:A6:B1 90-DD-5D-99-A6-B1
[NEW] Device 68:F0:BE:6E:30:08 68-F0-BE-6E-30-08
[NEW] Device 47:5D:98:F1:1F:BE 47-5D-98-F1-1F-BE
[NEW] Device 49:DC:39:A2:A4:30 49-DC-39-A2-A4-30
[NEW] Device 79:63:BF:F8:B4:B9 79-63-BF-F8-B4-B9
[NEW] Device 52:8A:92:02:9D:06 52-8A-92-02-9D-06
[NEW] Device 3D:58:1F:72:A0:A1 3D-58-1F-72-A0-A1
[NEW] Device 55:B2:67:D1:9D:0F 55-B2-67-D1-9D-0F
[NEW] Device 3D:D7:B0:3B:E0:EF 3D-D7-B0-3B-E0-EF
[NEW] Device 47:31:86:9E:B9:8B LE-Bose Color II SoundLink
[NEW] Device D0:03:4B:5A:D3:6A D0-03-4B-5A-D3-6A
[NEW] Device 5C:62:8B:2D:8E:AE 5C-62-8B-2D-8E-AE
[NEW] Device 74:EA:B6:42:8A:A8 74-EA-B6-42-8A-A8
[Lab3ABTmod]#
```

Figure 3: Bluetooth Scan begin in Raspberry Pi.



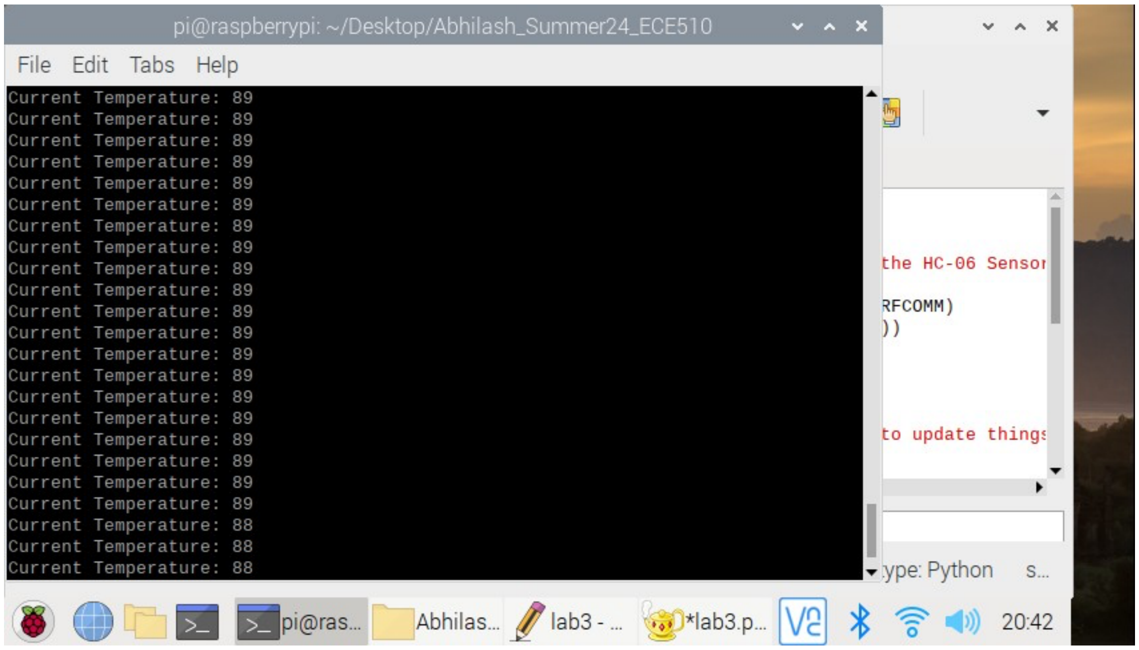
```
[CHG] Device D8:E3:5E:7A:89:27 UUIDs: 0000110e-0000-1000-8000-00805f9b34fb
[CHG] Device D8:E3:5E:7A:89:27 UUIDs: 00001200-0000-1000-8000-00805f9b34fb
[CHG] Device D8:E3:5E:7A:89:27 UUIDs: 00000000-0000-0000-0000-000000000000
[CHG] Device D8:E3:5E:7A:89:27 UUIDs: 0000feb9-0000-1000-8000-00805f9b34fb
[CHG] Device D8:E3:5E:7A:89:27 ManufacturerData Key: 0x00c4
[CHG] Device D8:E3:5E:7A:89:27 ManufacturerData Value:
  07 34 14 12 16 fd 80                              .4.....
[CHG] Device 67:D9:1B:77:8B:77 RSSI: -78
[CHG] Device D8:E3:5E:7A:89:27 RSSI: -65
[Lab3ABTmod]# scan off
[CHG] Device 5C:62:8B:2D:8E:AE RSSI is nil
[DEL] Device 5C:62:8B:2D:8E:AE 5C-62-8B-2D-8E-AE
[CHG] Device C4:23:60:5E:D0:BB TxPower is nil
[CHG] Device C4:23:60:5E:D0:BB RSSI is nil
[CHG] Device 22:22:6E:91:BA:65 RSSI is nil
[CHG] Device 00:22:02:01:02:F6 RSSI is nil
[CHG] Device D8:E3:5E:7A:89:27 TxPower is nil
[CHG] Device D8:E3:5E:7A:89:27 RSSI is nil
[CHG] Device 46:AE:F2:28:89:11 RSSI is nil
[CHG] Device 12:62:E5:CE:D4:43 TxPower is nil
[CHG] Device 12:62:E5:CE:D4:43 RSSI is nil
[CHG] Device 67:D9:1B:77:8B:77 RSSI is nil
[CHG] Device 46:46:C3:B2:2F:0C TxPower is nil
```

Figure 4: Bluetooth Scan End in Raspberry Pi.



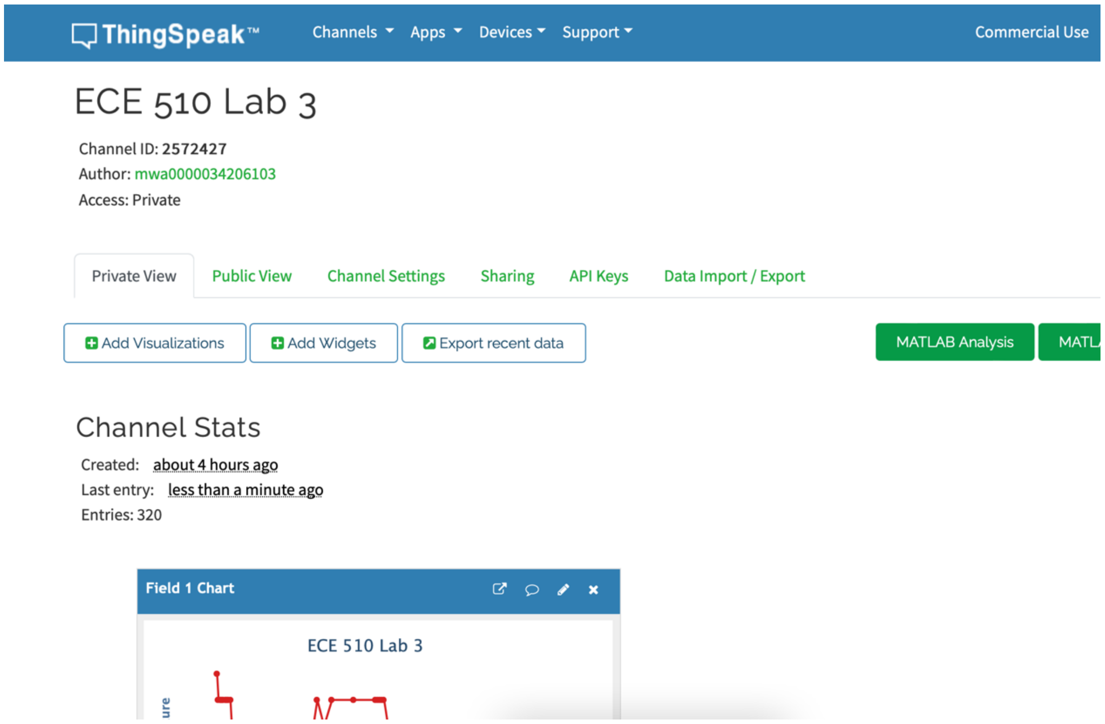
```
pi@raspberrypi:~ $ bluetoothctl
Agent registered
[Lab3ABTmod]# paired-devices
Device 00:22:02:01:02:F6 Lab3ABTmod
[Lab3ABTmod]#
```

Figure 5: Paired connection between Raspberry Pi and HC-06 Module with Arduino UNO.



```
Current Temperature: 89
Current Temperature: 89
Current Temperature: 89
Current Temperature: 89
Current Temperature: 89
Current Temperature: 89
Current Temperature: 89
Current Temperature: 89
Current Temperature: 89
Current Temperature: 89
Current Temperature: 89
Current Temperature: 89
Current Temperature: 89
Current Temperature: 89
Current Temperature: 89
Current Temperature: 89
Current Temperature: 89
Current Temperature: 89
Current Temperature: 89
Current Temperature: 89
Current Temperature: 88
Current Temperature: 88
Current Temperature: 88
```

Figure 6: Temperature Display on Raspberry Pi Terminal.


ThingSpeak™ Channels Apps Devices Support Commercial Use

ECE 510 Lab 3

Channel ID: 2572427
Author: mwa0000034206103
Access: Private

Private View | Public View | Channel Settings | Sharing | API Keys | Data Import / Export

Add Visualizations | Add Widgets | Export recent data | MATLAB Analysis | MATL

Channel Stats
Created: about 4 hours ago
Last entry: less than a minute ago
Entries: 320

Figure 7:ThingSpeak Private View.

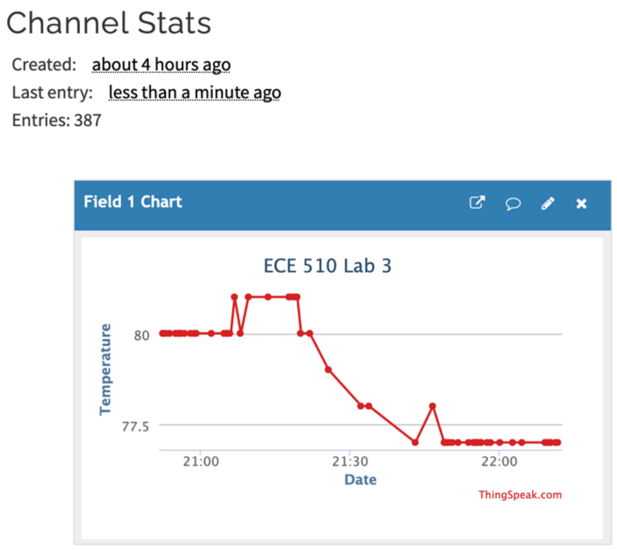

Channel Stats
Created: about 4 hours ago
Last entry: less than a minute ago
Entries: 387

Figure 8:ThingSpeak Graph.

The demo video is available at this [Link](#).

A. Discussion

1. **Explanation of the Temperature Formula Used**

To convert the raw analog-to-digital converter (ADC) result from a temperature sensor into a temperature in degrees Celsius, use the formula tempC = raw_sensor_value * 5 / 1024 * 100. The parts are broken out as follows:

- **raw_sensor_value**: This is the ADC-read integer value. The analogue voltage from the LM35 sensor is converted to a digital value by the Arduino's ADC. The Arduino Uno's ADC outputs a value between 0 and 1023 because it has a 10-bit resolution.
- **5**: The ADC's reference voltage, or V_ref. Usually, an Arduino Uno requires 5 volts for this.
- **1024**: Given that it is a 10-bit ADC, this is the ADC resolution (2^10).
- **100**: A voltage that is linearly proportional to the temperature in Celsius is produced by the LM35. It produces 10 mV per degree Celsius specifically. The factor 100 is used to convert volts to millivolts and then back to temperature (since 1V = 1000mV and 10mV = 1°C, 1000mV/10mV = 100).

Combining this:

First, the ADC value (raw_sensor_value) is multiplied by the reference voltage and divided by the ADC resolution to get the voltage: $voltage = \frac{raw\_sensor\_value\ X\ 5}{1024}$.

Next, this voltage is multiplied by 100 to get the temperature in Celsius: $tempC = \left(\frac{raw\_sensor\_value\ X\ 5}{1024}\right) X\ 100$.

As a result, the equation successfully translates the ADC value to the equivalent LM35 sensor temperature reading.

2. **Replacement of DHT22 sensor with either LM35 or DHT11**

- Using LM35:

Hardware Connections: Attach the LM35 sensor's GND to ground, VCC to the Arduino's 5V pin, and output to one of the Arduino's analogue input pins (such as A0).

Code Modifications: The LM35 sensor's analogue value must be read by the code, which then uses the formula tempC = raw_sensor_value * 5 / 1024 * 100 to convert it to temperature.

Library: To obtain the sensor value, simply use Arduino's analogRead function; no special library is required.

- Using DHT 11

Hardware Connections: Attach the data pin to an Arduino digital input pin, GND to ground, and VCC to 5V, much like on the DHT22.

Code Modifications: Include the DHT library (DHT.h) in the code. Initializing the sensor as DHT dht (DHTPIN, DHTTYPE) is required, with DHTTYPE specified as DHT11 rather than DHT22.

Library: Make sure the DHT library is configured for DHT11 and include it. This entails switching the DHT22 to DHT11 setup and any associated special calls.

Data handling: In comparison to the DHT22, the DHT11 sensor is less accurate and uses a different sample rate. If necessary, modify your code to account for these differences.

- For LM35: Use a new conversion formula and switch from digital to analogue reading.
- For DHT11: Make sure the right library and initialization are used and modify the sensor type in the code.

3. **Interfacing two Arduino boards using Bluetooth**

To connect two Arduino boards using Bluetooth, the following extra hardware is usually required:

- Bluetooth Modules: Two Bluetooth modules (HC-05 or HC-06, for example) are used. We'll configure one module to be the master and the other as the slave. The HC-06 is a slave-only device; however, the HC-05 can be set up as a master or slave.
- Level Shifters: To guarantee correct communication between the Bluetooth modules and the Arduino boards, you may want level shifters if you are utilizing Arduino boards that run at various logic levels (for example, one at 5V and another at 3.3V).
- Power Supply: Make sure there is a steady power source for the Arduino boards and the Bluetooth modules. This could entail making sure the power source can support the combined load or utilizing different power supplies for every board.
- Connection cables/wires: To link the Bluetooth modules to the Arduino boards, you'll need the right cables and wires. These normally include VCC, GND, TX, and RX connectors.

4. **Using Bluetooth Low Energy (BLE) in place of Classic Bluetooth**

The following modifications and variations would apply if the interface were to be implemented using a Bluetooth Low Energy (BLE) device rather than a traditional Bluetooth device:

- BLE Modules: Traditional Bluetooth modules like the HC-06 would be replaced with BLE modules like the HM-10 or nRF52. Compared to traditional Bluetooth, BLE is intended for different use cases and lower power consumption.
- Libraries and Tools: For BLE communication in Python, you would need libraries like bleak instead of pybluez, which is used for classic Bluetooth. A Python library called bleak makes BLE operations easier.
- Device Identification and Linking:
    - Classic Bluetooth: Bluetoothctl and commands like pair, connect, etc. are used to scan and link devices.

    - BLE: BLE-specific scanning functions are used to find BLE devices. BleakClient is used to establish connections, and BleakScanner is used to find devices in bleak.
- GATT Protocol: BLE communicates over the Generic Attribute Profile (GATT). This includes features and services for writing and reading data.
    - Classic Bluetooth: Usually, a straightforward serial interface is used for communication.
    - BLE: You must manage features and services. For instance, to submit data or get notifications, you might need to subscribe to or write to a characteristic.
- Power Consumption: BLE is intended to use less power, which makes it appropriate for devices that run on batteries. In general, traditional Bluetooth uses more energy.

References

[1] "Experiment 03," Lab Manual.

[2] staff, "LM35 Precision Centigrade Temperature Sensors," Texas Instruments, 2017.

[3] l. ada, "Adafruit | Overview," Adafruit, 07 June 2024. [Online]. Available: https://learn.adafruit.com/dht. [Accessed June 2024].

[4] B. Henrik, "Bleak," 2020. [Online]. Available: https://bleak.readthedocs.io/en/latest/.

[5] J. Fraden, Handbook of modern sensors, American Institute of Physics, 1994.