Design Lab Experiment No. 3:

*Temperature Sensing System using Bluetooth*

By: Alan Palayil

Lab Partner: Individual

Instructor: Prof. Jafar Saniie, Prof. Won-Jae Yi

TA: Mikhail Gromov

ECE 442 Internet of Things and Cyber Physical Systems

Date: 06-03-2022

**Acknowledgment**: I acknowledge all works including figures, codes and writings belong to me and/or persons who are referenced. I understand if any similarity in the code, comments, customized program behavior, report writings and/or figures are found, both the helper (original work) and the requestor (duplicated/modified work) will be called for academic disciplinary action.

Signature: \_\_\_\_Alan Biju Palayil\_\_\_\_\_\_\_\_\_\_

**I. Introduction**

**A. Purpose**

The purpose of this experiment is to get familiar with the temperature sensing system using DHT11 temperature sensor wired to the Arduino Uno board which transmits the data from the sensor to the Pi via HC-06 Bluetooth module and uploads the data to ThinkSpeak to visualize the data through the internet. This experiment shows how Arduino communicates via the Bluetooth with the Raspberry Pi and the process to upload data to the cloud.

**B. Background**

The Arduino Uno and Raspberry Pi are the key components of this lab. Using Arduino to program the board and sensors and python script for the Raspberry Pi. The DHT11 temperature sensor is connected to the Arduino where it can transmit the data over to the Pi using Bluetooth. The temperature sensor monitors the ambient temperature and humidity and measures the data in Centigrade. The HC-06 module can be configured using AT commands over serial connection. The HC-06 Bluetooth module acts as a slave device, the connection is made by pairing it with the Pi using RFCOMM. ThinkSpeak is an IoT open-sourced application and API to retrieve and store data from various sensors using HTTP over the internet via LAN.

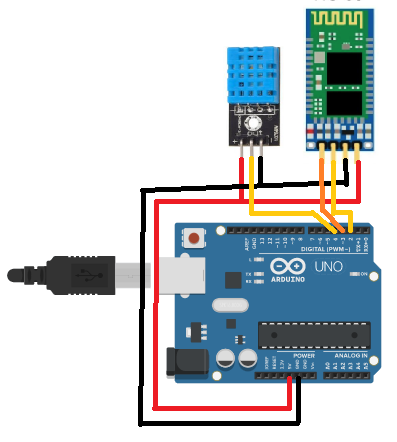
**II. Lab Procedure and Equipment List**

1. **Equipment**
   * 1 x Raspberry Pi 3 single-board computer
   * 1 x Arduino UNO board and USB cable
   * 1 x HC-06 Bluetooth transceiver
   * 1 x LM35 Ambient Temperature Sensor or DHT11 Temperature and Humidity Sensor
   * 1 x 5K or 10K Ohms Pull Up Resistor (for DHT11 Sensor)
   * 1 x Breadboard
2. **Procedure**

\*Refer to Lab Manual for detailed Procedure\*

**III. Results and Analysis**

1. **Results**

**A picture containing text, indoor, computer, electronics

Description automatically generated**

*Schematics for the DHT11 sensor and HC-06 module with the Arduino*

Graphical user interface, text, application

Description automatically generated

*COM3 on Arduino*

Text

Description automatically generatedText

Description automatically generated

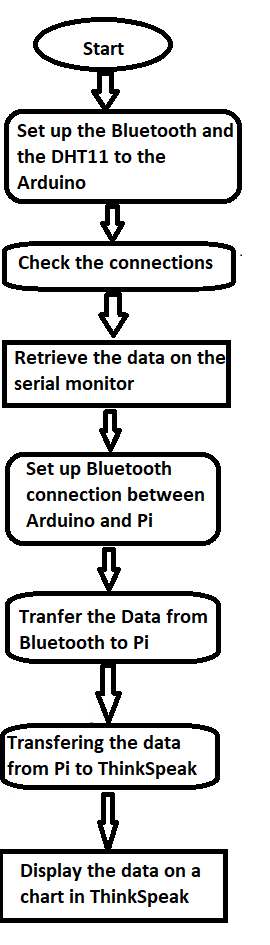
*Pairing and Python script on Raspberry PI*

Graphical user interface, text, application

Description automatically generated

*ThinkSpeak channel for Temperature Graph*

1. **Answers to Discussion Questions**
   1. See Appendix for the fully commented code.
   2. *Flowchart of the overall Temperature Sensing System*



* 1. The datasheet of Atmega states that its ADC has a definition of 10-bits (1024 values), so a formula is used to convert the retrieved integer from Analog to Digital. The temperature is calculated in Celsius using the given formula:

temperature\_celsius= ((((raw\_sensor\_value)/1024)\*5000)/10);

Then it is converted to Fahrenheit by:

temperature\_fahrenheit=(((temperature\_celsius\*9)/5)+32);

And the calculated data is output on the serial monitor COM3.

* 1. The Raspberry Pi is first paired to the Bluetooth module of Arduino using the bluetoothctl command. After scanning for the Bluetooth module and pairing with it using it is MAC address and PIN. Then Python is set up on Pi using the following commands:

Sudo apt update

Sudo apt install Bluetooth libbluetooth-dev …

An account is created in ThinkSpeak, and the channel ID and API key is obtained and put in the python script. We execute the code and get the results in the Pi as well as it being uploaded to ThinkSpeak using the ID and API.

* 1. The Bluetooth Low Energy (BLE) devices uses BLE libraries to set up the BLE module in Arduino and for the Raspberry we need to install the pybluez and bluepy libraries for python along with -e .[ble] for the supporting files.

An example code for the python script in Raspberry:

# bluetooth low energy scan

from bluetooth.ble import DiscoveryService

service = DiscoveryService()

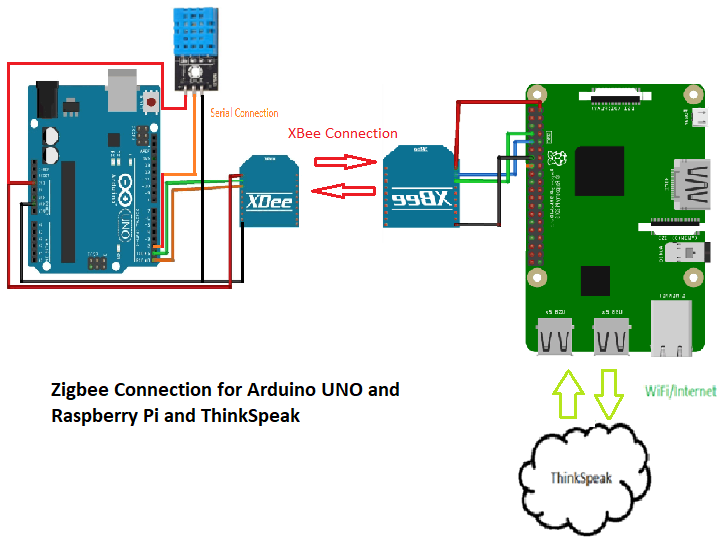
devices = service.discover(2)

for address, name in devices.items():

print("name: {}, address: {}".format(name, address))

* 1. ZigBee is a communication device that is used for the data transfer between the controllers, computers, systems, etc. with a serial port. It works with low power consumption devices and has a transmission of 10m to 100m in a line-of-sight. A Xbee module is connected to the Arduino board and the other Xbee module is connected to the Raspberry Pi. The modules are then set up using proper code to establish the communication.

The connection set up for the Arduino will be like the process for HC-06 while for the Raspberry Pi depending on the Xbee module can be connected either via the USB port or the serial pins. The Xbee modules are set up using XCTU software. The XCTU’s console terminal is used to give commands and set up the connections between the Xbee modules.



**IV. Conclusion**

This lab helped us read real-time data and display it on the internet. The temperature sensing system using DHT11 temperature sensor wired to the Arduino Uno, the data from the sensors transmitted via Bluetooth using the HC-06 module to Raspberry Pi and upload the received data to ThinkSpeak to visualize the sensor data graphically online.

**References**

[1] Experiment 1 Lab Manual

[2] Lecture Note 3 of ECE 442/510

**Appendix**

1. *Lab3.py Raspberry Pi Program*

import bluetooth # needed for bluetooth communication

import thingspeak # needed for thingspeak

bluetooth\_addr = "00:20:12:08:2A:D8" # The address from the HC-06 sensor

bluetooth\_port = 1 # Channel 1 for RFCOMM

bluetoothSocket = bluetooth.BluetoothSocket (bluetooth.RFCOMM)

bluetoothSocket.connect((bluetooth\_addr,bluetooth\_port))

#thingspeak information

channel\_id = 1758168 # channel ID from your Thingspeak channel

key = "5CB3I11C59J5LTMY" # obtain from Thingspeak

url = 'https://api.thinkspeak.com/update' # default URL to update Thingspeak

ts = thingspeak.Channel(channel\_id, key, url)

while 1:

try:

received\_data = bluetoothSocket.recv(1024)

temperature = int.from\_bytes(received\_data,byteorder='big')

print("Current Temperature: %d" % temperature)

thingspeak\_field1 = {"field1": temperature}

ts.update(thingspeak\_field1) # update thingspeak

except KeyboardInterrupt:

print("keyboard interrupt detected")

break

bluetoothSocket.close()

1. *Code on Arduino*

#include <SoftwareSerial.h>

#include "dht.h"

SoftwareSerial mySerial(5, 6); // RX, TX

#define DHT\_Sensor 7

dht sensor;

void setup() {

// Open serial communications and wait for port to open:

Serial.begin(9600);

mySerial.begin(38400);

}

void loop() { // run over and over

sensor.read11(DHT\_Sensor);

float tc=sensor.temperature;

float tf=tc\*1.8+32;

Serial.print(tc);

Serial.print("C (");

Serial.print(tf);

Serial.print("F)\t");

mySerial.write(tf);

Serial.println("Bluetooth Transmission Complete");

Serial.flush();

delay(1000);

}

1. *dht.h*

*//*

*// FILE: dht.h*

*// AUTHOR: Rob Tillaart*

*// VERSION: 0.1.29*

*// PURPOSE: DHT Temperature & Humidity Sensor library for Arduino*

*// URL: http://arduino.cc/playground/Main/DHTLib*

*//*

*// HISTORY:*

*// see dht.cpp file*

*//*

*#ifndef dht\_h*

*#define dht\_h*

*#if ARDUINO < 100*

*#include <WProgram.h>*

*#include <pins\_arduino.h> // fix for broken pre 1.0 version - TODO TEST*

*#else*

*#include <Arduino.h>*

*#endif*

*#define DHT\_LIB\_VERSION "0.1.29"*

*#define DHTLIB\_OK 0*

*#define DHTLIB\_ERROR\_CHECKSUM -1*

*#define DHTLIB\_ERROR\_TIMEOUT -2*

*#define DHTLIB\_ERROR\_CONNECT -3*

*#define DHTLIB\_ERROR\_ACK\_L -4*

*#define DHTLIB\_ERROR\_ACK\_H -5*

*#define DHTLIB\_DHT11\_WAKEUP 18*

*#define DHTLIB\_DHT\_WAKEUP 1*

*#define DHTLIB\_DHT11\_LEADING\_ZEROS 1*

*#define DHTLIB\_DHT\_LEADING\_ZEROS 6*

*// max timeout is 100 usec.*

*// For a 16 Mhz proc 100 usec is 1600 clock cycles*

*// loops using DHTLIB\_TIMEOUT use at least 4 clock cycli*

*// so, 100 us takes max 400 loops*

*// so, by dividing F\_CPU by 40000 we "fail" as fast as possible*

*#ifndef F\_CPU*

*#define DHTLIB\_TIMEOUT 1000 // ahould be approx. clock/40000*

*#else*

*#define DHTLIB\_TIMEOUT (F\_CPU/40000)*

*#endif*

*class dht*

*{*

*public:*

*dht() { \_disableIRQ = false; };*

*// return values:*

*// DHTLIB\_OK*

*// DHTLIB\_ERROR\_CHECKSUM*

*// DHTLIB\_ERROR\_TIMEOUT*

*// DHTLIB\_ERROR\_CONNECT*

*// DHTLIB\_ERROR\_ACK\_L*

*// DHTLIB\_ERROR\_ACK\_H*

*int8\_t read11(uint8\_t pin);*

*int8\_t read(uint8\_t pin);*

*int8\_t read12(uint8\_t pin);*

*inline int8\_t read21(uint8\_t pin) { return read(pin); };*

*inline int8\_t read22(uint8\_t pin) { return read(pin); };*

*inline int8\_t read33(uint8\_t pin) { return read(pin); };*

*inline int8\_t read44(uint8\_t pin) { return read(pin); };*

*inline int8\_t read2301(uint8\_t pin) { return read(pin); };*

*inline int8\_t read2302(uint8\_t pin) { return read(pin); };*

*inline int8\_t read2303(uint8\_t pin) { return read(pin); };*

*inline int8\_t read2320(uint8\_t pin) { return read(pin); };*

*inline int8\_t read2322(uint8\_t pin) { return read(pin); };*

*bool getDisableIRQ() { return \_disableIRQ; };*

*void setDisableIRQ(bool b ) { \_disableIRQ = b; };*

*float humidity;*

*float temperature;*

*private:*

*uint8\_t bits[5]; // buffer to receive data*

*int8\_t \_readSensor(uint8\_t pin, uint8\_t wakeupDelay, uint8\_t leadingZeroBits);*

*bool \_disableIRQ;*

*};*

*#endif*

*//*

*// END OF FILE*

*//*

1. *dht.cpp*

*//*

*// FILE: dht.cpp*

*// AUTHOR: Rob Tillaart*

*// VERSION: 0.1.29*

*// PURPOSE: DHT Temperature & Humidity Sensor library for Arduino*

*// URL: http://arduino.cc/playground/Main/DHTLib*

*//*

*// HISTORY:*

*// 0.1.29 2018-09-02 fix negative temperature DHT12 - issue #111*

*// 0.1.28 2018-04-03 refactor*

*// 0.1.27 2018-03-26 added \_disableIRQ flag*

*// 0.1.26 2017-12-12 explicit support for AM23XX series and DHT12*

*// 0.1.25 2017-09-20 FIX https://github.com/RobTillaart/Arduino/issues/80*

*// 0.1.24 2017-07-27 FIX https://github.com/RobTillaart/Arduino/issues/33 double -> float*

*// 0.1.23 2017-07-24 FIX https://github.com/RobTillaart/Arduino/issues/31*

*// 0.1.22 undo delayMicroseconds() for wakeups larger than 8*

*// 0.1.21 replace delay with delayMicroseconds() + small fix*

*// 0.1.20 Reduce footprint by using uint8\_t as error codes. (thanks to chaveiro)*

*// 0.1.19 masking error for DHT11 - FIXED (thanks Richard for noticing)*

*// 0.1.18 version 1.16/17 broke the DHT11 - FIXED*

*// 0.1.17 replaced micros() with adaptive loopcount*

*// removed DHTLIB\_INVALID\_VALUE*

*// added DHTLIB\_ERROR\_CONNECT*

*// added DHTLIB\_ERROR\_ACK\_L DHTLIB\_ERROR\_ACK\_H*

*// 0.1.16 masking unused bits (less errors); refactored bits[]*

*// 0.1.15 reduced # micros call 2->1 in inner loop.*

*// 0.1.14 replace digital read with faster (~3x) code => more robust low MHz machines.*

*//*

*// 0.1.13 fix negative temperature*

*// 0.1.12 support DHT33 and DHT44 initial version*

*// 0.1.11 renamed DHTLIB\_TIMEOUT*

*// 0.1.10 optimized faster WAKEUP + TIMEOUT*

*// 0.1.09 optimize size: timeout check + use of mask*

*// 0.1.08 added formula for timeout based upon clock speed*

*// 0.1.07 added support for DHT21*

*// 0.1.06 minimize footprint (2012-12-27)*

*// 0.1.05 fixed negative temperature bug (thanks to Roseman)*

*// 0.1.04 improved readability of code using DHTLIB\_OK in code*

*// 0.1.03 added error values for temperature and humidity when read failed*

*// 0.1.02 added error codes*

*// 0.1.01 added support for Arduino 1.0, fixed typos (31/12/2011)*

*// 0.1.00 by Rob Tillaart (01/04/2011)*

*//*

*// inspired by DHT11 library*

*//*

*// Released to the public domain*

*//*

*#include "dht.h"*

*/////////////////////////////////////////////////////*

*//*

*// PUBLIC*

*//*

*int8\_t dht::read11(uint8\_t pin)*

*{*

*// READ VALUES*

*if (\_disableIRQ) noInterrupts();*

*int8\_t result = \_readSensor(pin, DHTLIB\_DHT11\_WAKEUP, DHTLIB\_DHT11\_LEADING\_ZEROS);*

*if (\_disableIRQ) interrupts();*

*// these bits are always zero, masking them reduces errors.*

*bits[0] &= 0x7F;*

*bits[2] &= 0x7F;*

*// CONVERT AND STORE*

*humidity = bits[0]; // bits[1] == 0;*

*temperature = bits[2]; // bits[3] == 0;*

*// TEST CHECKSUM*

*uint8\_t sum = bits[0] + bits[1] + bits[2] + bits[3];*

*if (bits[4] != sum)*

*{*

*return DHTLIB\_ERROR\_CHECKSUM;*

*}*

*return result;*

*}*

*int8\_t dht::read12(uint8\_t pin)*

*{*

*// READ VALUES*

*if (\_disableIRQ) noInterrupts();*

*int8\_t result = \_readSensor(pin, DHTLIB\_DHT11\_WAKEUP, DHTLIB\_DHT11\_LEADING\_ZEROS);*

*if (\_disableIRQ) interrupts();*

*// CONVERT AND STORE*

*humidity = bits[0] + bits[1] \* 0.1;*

*temperature = bits[2] + (bits[3] & 0x7F) \* 0.1;*

*if (bits[3] & 0x80) // negative temperature*

*{*

*temperature = -temperature;*

*}*

*// TEST CHECKSUM*

*uint8\_t sum = bits[0] + bits[1] + bits[2] + bits[3];*

*if (bits[4] != sum)*

*{*

*return DHTLIB\_ERROR\_CHECKSUM;*

*}*

*return result;*

*}*

*int8\_t dht::read(uint8\_t pin)*

*{*

*// READ VALUES*

*if (\_disableIRQ) noInterrupts();*

*int8\_t result = \_readSensor(pin, DHTLIB\_DHT\_WAKEUP, DHTLIB\_DHT\_LEADING\_ZEROS);*

*if (\_disableIRQ) interrupts();*

*// these bits are always zero, masking them reduces errors.*

*bits[0] &= 0x03;*

*bits[2] &= 0x83;*

*// CONVERT AND STORE*

*humidity = (bits[0]\*256 + bits[1]) \* 0.1;*

*temperature = ((bits[2] & 0x7F)\*256 + bits[3]) \* 0.1;*

*if (bits[2] & 0x80) // negative temperature*

*{*

*temperature = -temperature;*

*}*

*// TEST CHECKSUM*

*uint8\_t sum = bits[0] + bits[1] + bits[2] + bits[3];*

*if (bits[4] != sum)*

*{*

*return DHTLIB\_ERROR\_CHECKSUM;*

*}*

*return result;*

*}*

*/////////////////////////////////////////////////////*

*//*

*// PRIVATE*

*//*

*int8\_t dht::\_readSensor(uint8\_t pin, uint8\_t wakeupDelay, uint8\_t leadingZeroBits)*

*{*

*// INIT BUFFERVAR TO RECEIVE DATA*

*uint8\_t mask = 128;*

*uint8\_t idx = 0;*

*uint8\_t data = 0;*

*uint8\_t state = LOW;*

*uint8\_t pstate = LOW;*

*uint16\_t zeroLoop = DHTLIB\_TIMEOUT;*

*uint16\_t delta = 0;*

*leadingZeroBits = 40 - leadingZeroBits; // reverse counting...*

*// replace digitalRead() with Direct Port Reads.*

*// reduces footprint ~100 bytes => portability issue?*

*// direct port read is about 3x faster*

*uint8\_t bit = digitalPinToBitMask(pin);*

*uint8\_t port = digitalPinToPort(pin);*

*volatile uint8\_t \*PIR = portInputRegister(port);*

*// REQUEST SAMPLE*

*pinMode(pin, OUTPUT);*

*digitalWrite(pin, LOW); // T-be*

*if (wakeupDelay > 8) delay(wakeupDelay);*

*else delayMicroseconds(wakeupDelay \* 1000UL);*

*// digitalWrite(pin, HIGH); // T-go*

*pinMode(pin, INPUT);*

*uint16\_t loopCount = DHTLIB\_TIMEOUT \* 2; // 200uSec max*

*// while(digitalRead(pin) == HIGH)*

*while ((\*PIR & bit) != LOW )*

*{*

*if (--loopCount == 0)*

*{*

*return DHTLIB\_ERROR\_CONNECT;*

*}*

*}*

*// GET ACKNOWLEDGE or TIMEOUT*

*loopCount = DHTLIB\_TIMEOUT;*

*// while(digitalRead(pin) == LOW)*

*while ((\*PIR & bit) == LOW ) // T-rel*

*{*

*if (--loopCount == 0)*

*{*

*return DHTLIB\_ERROR\_ACK\_L;*

*}*

*}*

*loopCount = DHTLIB\_TIMEOUT;*

*// while(digitalRead(pin) == HIGH)*

*while ((\*PIR & bit) != LOW ) // T-reh*

*{*

*if (--loopCount == 0)*

*{*

*return DHTLIB\_ERROR\_ACK\_H;*

*}*

*}*

*loopCount = DHTLIB\_TIMEOUT;*

*// READ THE OUTPUT - 40 BITS => 5 BYTES*

*for (uint8\_t i = 40; i != 0; )*

*{*

*// WAIT FOR FALLING EDGE*

*state = (\*PIR & bit);*

*if (state == LOW && pstate != LOW)*

*{*

*if (i > leadingZeroBits) // DHT22 first 6 bits are all zero !! DHT11 only 1*

*{*

*zeroLoop = min(zeroLoop, loopCount);*

*delta = (DHTLIB\_TIMEOUT - zeroLoop)/4;*

*}*

*else if ( loopCount <= (zeroLoop - delta) ) // long -> one*

*{*

*data |= mask;*

*}*

*mask >>= 1;*

*if (mask == 0) // next byte*

*{*

*mask = 128;*

*bits[idx] = data;*

*idx++;*

*data = 0;*

*}*

*// next bit*

*--i;*

*// reset timeout flag*

*loopCount = DHTLIB\_TIMEOUT;*

*}*

*pstate = state;*

*// Check timeout*

*if (--loopCount == 0)*

*{*

*return DHTLIB\_ERROR\_TIMEOUT;*

*}*

*}*

*// pinMode(pin, OUTPUT);*

*// digitalWrite(pin, HIGH);*

*return DHTLIB\_OK;*

*}*

*//*

*// END OF FILE*

*//*