GL5516

* Run script to register MAC address for esp32
  + Make sure all devices (raspberry pi, esp32, and personal computer) are connected to yale data
    - ssid: “yale wireless”
    - No password!
* Getting the light sensor data:
  + Followed the wiring for the sensor circuit
    - <https://esp32io.com/tutorials/esp32-light-sensor>
  + Flashed the code to read the input from the Arduino IDE
    - The name of the sketch for the light sensor is: “”

/\*

\* This ESP32 code is created by esp32io.com

\*

\* This ESP32 code is released in the public domain

\*

\* For more detail (instruction and wiring diagram), visit https://esp32io.com/tutorials/esp32-light-sensor

\*/

#define LIGHT\_SENSOR\_PIN 36 // ESP32 pin GIOP36 (ADC0)

void setup() {

// initialize serial communication at 9600 bits per second:

Serial.begin(9600);

}

void loop() {

// reads the input on analog pin (value between 0 and 4095)

int analogValue = analogRead(LIGHT\_SENSOR\_PIN);

Serial.print("Analog Value = ");

Serial.print(analogValue); // the raw analog reading

// We'll have a few threshholds, qualitatively determined

if (analogValue < 40) {

Serial.println(" => Dark");

} else if (analogValue < 800) {

Serial.println(" => Dim");

} else if (analogValue < 2000) {

Serial.println(" => Light");

} else if (analogValue < 3200) {

Serial.println(" => Bright");

} else {

Serial.println(" => Very bright");

}

delay(500);

}

* Use UDP to send info wirelessly
  + we “ll simply create the udp server, which will be able to read and write on the localPort port.
  + <https://www.aranacorp.com/en/udp-communication-between-raspberry-pi-and-esp32/#:~:text=When%20devices%20are%20connected%20to,UDP%20protocol%20between%20the%20two>.
  + The [socket](https://wiki.python.org/moin/UdpCommunication) library can be used to open a connection between two devices.
* First, let’s create the UDP server on the raspberry pi
  + Set up new folder after ssh-ing into the raspberry pi
  + Then create a new file and put in this code to create the UDP server:
    - #!/usr/bin/env python
    - # -\*- coding: utf-8 -\*-
    - #Libraries
    - import socket #https://wiki.python.org/moin/UdpCommunication
    - #Parameters
    - localPort=8888
    - bufferSize=1024
    - #Objects
    - sock = socket.socket(socket.AF\_INET,socket.SOCK\_DGRAM) ## Internet,UDP
    - # function init
    - def init():
    - sock.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEPORT, 1)
    - sock.setsockopt(socket.SOL\_SOCKET, socket.SO\_BROADCAST, 1) #enable broadcasting mode
    - sock.bind(('', localPort))
    - print("UDP server : {}:{}".format(get\_ip\_address(),localPort))
    - # function main
    - def main():
    - while True:
    - data, addr = sock.recvfrom(1024) # get data
    - print("received message: {} from {}\n".format(data,addr))
    - sock.sendto("RPi received OK",addr) # write data
    - # function get\_ip\_address
    - def get\_ip\_address():
    - """get host ip address"""
    - ip\_address = '';
    - s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)
    - s.connect(("8.8.8.8",80))
    - ip\_address = s.getsockname()[0]
    - s.close()
    - return ip\_address
    - if \_\_name\_\_ == '\_\_main\_\_':
    - init()
    - main()
  + After running this code ^, the program outputs the following:
    - UDP server: 172.29.131.13:8888
      * So, the UDP server running on the raspberry pi has:
        + Ip address: 172.29.131
        + Port: 888
* Then make the client-side code to run on the esp32
* Testing the connection:
  + First run the esp32\_wifi.py code to make sure the server is already running on the raspberry pi
  + Then run the esp32 Arduino code - we should see both IP addresses and ports printed, plus the light sensor data
  + Debugging: TypeError: a bytes-like object is required not "str" send data wireless udp
    - This is a type error: do “received data”.encode(‘utf-8’) to change from string type to bytes in the raspberry pi program
    - This is the data
* Then add the code to measure the amount of light that the sensor measures from the esp32 side
  + #define LIGHT\_SENSOR\_PIN 36 // ESP32 pin GIOP36 (ADC0)
  + int analogValue = analogRead(LIGHT\_SENSOR\_PIN);
  + Put this
* Make a visualization aspect that reads in the light sensor data and have it change something visually or sonically
  + Using pygame module
    - Opens a window with a blue dot that changes size depending on amount of light the sensor receives
  + Added the following lines:

Final client-side ESP32 code incorporating light sensor, client-side UDP connection, and pygame (for visualization):

* + #include <WiFi.h>
  + #include <WiFiUdp.h>
  + WiFiUDP udp;
  + #define LIGHT\_SENSOR\_PIN 36 // ESP32 pin GIOP36 (ADC0)
  + char packetBuffer[255];
  + unsigned int localPort = 9999;
  + char \*serverip = "172.29.131.13";
  + unsigned int serverport = 8888;
  + const char \*ssid = "yale wireless";
  + void setup() {
  + Serial.begin(115200);
  + // Connect to Wifi network.
  + WiFi.begin(ssid);
  + while (WiFi.status() != WL\_CONNECTED) {
  + delay(500); Serial.print(F("."));
  + }
  + udp.begin(localPort);
  + Serial.printf("UDP Client : %s:%i \n", WiFi.localIP().toString().c\_str(), localPort);
  + }
  + void loop() {
  + // reads the input on analog pin (value between 0 and 4095)
  + int analogValue = analogRead(LIGHT\_SENSOR\_PIN);
  + int packetSize = udp.parsePacket();
  + if (packetSize) {
  + Serial.print(" Received packet from : "); Serial.println(udp.remoteIP());
  + int len = udp.read(packetBuffer, 255);
  + Serial.printf("Data : %s\n", packetBuffer);
  + Serial.println();
  + }
  + delay(500);
  + Serial.print("[Client Connected] "); Serial.println(WiFi.localIP());
  + udp.beginPacket(serverip, serverport);
  + char buf[30];
  + unsigned long testID = millis();
  + sprintf(buf, "ESP32 send millis: %lu", testID);
  + sprintf(buf, "ESP32 send light data: %lu", analogValue);
  + Serial.println(analogValue);
  + udp.printf(buf);
  + udp.endPacket();
  + }

UDP client code (on raspberry pi)

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

#Libraries

import socket #https://wiki.python.org/moin/UdpCommunication

import pygame

pygame.init()

# set up the drawing window

screen = pygame.display.set\_mode([500, 500])

#Parameters

localPort=8888

bufferSize=1024

#Objects

sock = socket.socket(socket.AF\_INET,socket.SOCK\_DGRAM) ## Internet,UDP

# function init

def init():

sock.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEPORT, 1)

sock.setsockopt(socket.SOL\_SOCKET, socket.SO\_BROADCAST, 1) #enable broadcasting mode

sock.bind(('', localPort))

print("UDP server : {}:{}".format(get\_ip\_address(),localPort))

# function main

def main():

while True:

data, addr = sock.recvfrom(1024) # get data

data\_int = int(data[23:])

screen.fill((255,255,255))

# draw a solid blue circle

pygame.draw.circle(screen, (0,0,255), (250,250), data\_int//10)

# flip the display

pygame.display.flip()

print(addr)

print(data)

print("received message: {} from {}\n".format(data,addr))

sock.sendto("received ok".encode('utf-8'), addr) # write data

# function get\_ip\_address

def get\_ip\_address():

"""get host ip address"""

ip\_address = '';

s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

s.connect(("8.8.8.8",80))

ip\_address = s.getsockname()[0]

s.close()

return ip\_address

if \_\_name\_\_ == '\_\_main\_\_':

init()

main()