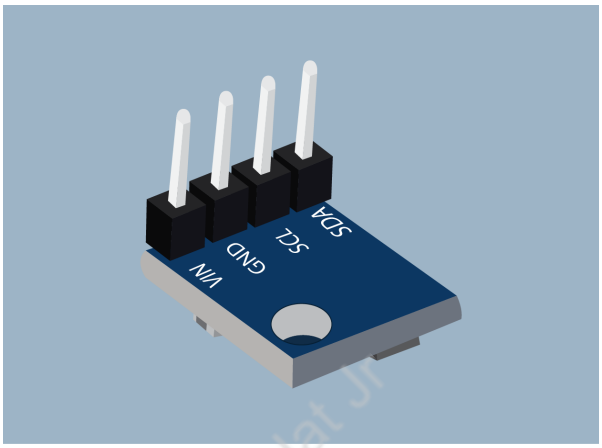
ANALOG INPUTS AND SENSORS



What is our GOAL for this CLASS?

In this class, we learned about analog inputs and used potentiometers to create analog waveforms. We also learned about DHT11 sensors and used these to measure pressure and temperature.

What did we ACHIEVE in the class TODAY?

- We learned about the basics of **analog input & potentiometer**.
- We learned about the **DTH11 sensor**
- We learned about **Circuit design**.
- We learned to read **the temperature and humidity value from DHT11** print it to the serial Monitor and checked the same on the graph.

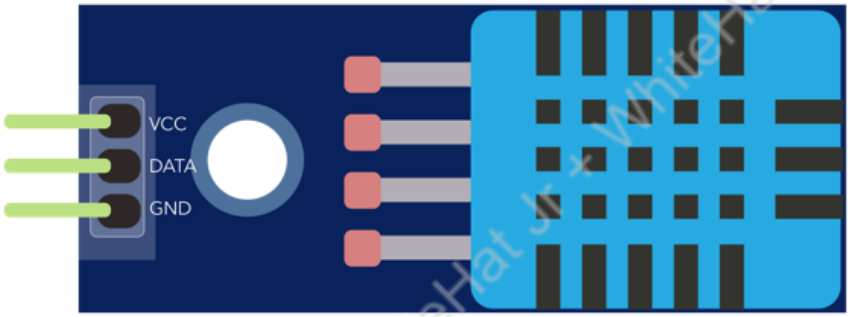
Which CONCEPTS/ CODING BLOCKS did we cover today?

- We used **Potentiometer**
    - A potentiometer is a 3 terminal device in which the resistance is manually varied to control the flow of electric current.
    - A potentiometer usually has 3 pins:
        - **VCC pin**: Connect this pin to VCC (5V or 3.3v)
        - **GND pin**: connect this pin to GND (0V)

WhiteHat Jr + WhiteHat Jr

    - **DATA pin**: the pin is used to communicate between the potentiometer and ESP32, it outputs the voltage to ESP32's input pin.

- We used **DTH11 sensors** to measure pressure and humidity. DHT11 include 4 pins:
    - **GND pin**: connect this pin to GND (0V)
    - **VCC pin**: connect this pin to VCC (3.3V)
    - **DATA pin**: the pin is used to communicate between the sensor and ESP32, outputs the voltage to ESP32's input pin.



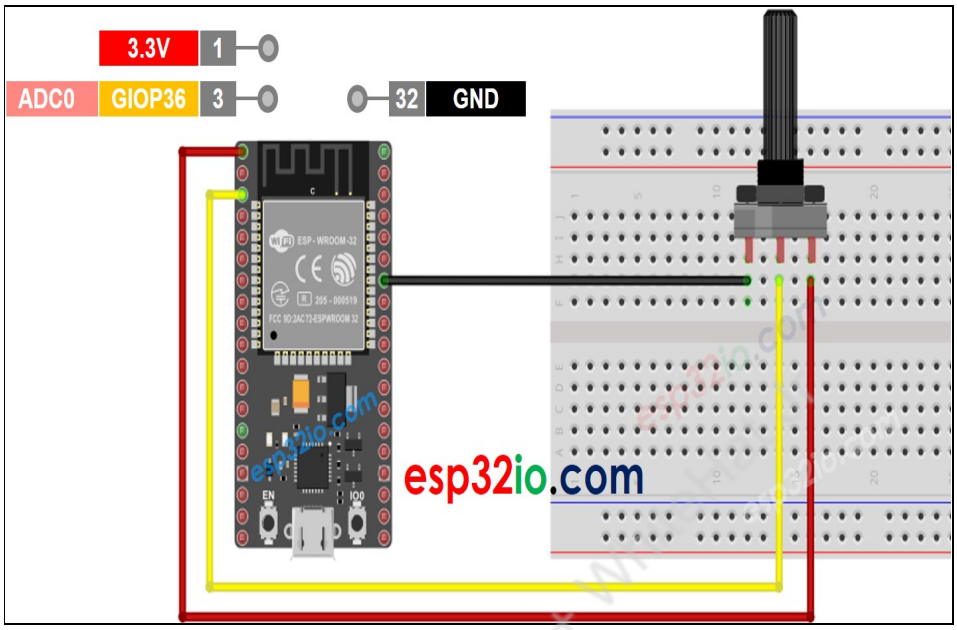
WhiteHat Jr + WhiteHat Jr

How did we DO the activities?

1. Gather the material from the IoT kit:
    - 1 x ESP32
    - 1 x USB Cable
    - 1 x Breadboard
    - 4 x Jumper wires
    - 1 x Potentiometer

2. Connections for **Circuit** Diagram
    - Supply **positive(VCC (+ve))** from the ESP 32 to breadboard terminal
    - Supply **negative(GND (-ve))** from the ESP 32 to breadboard negative terminal
    - Insert **potentiometer** into the breadboard
    - Keep track of all three pins, first and last pin of potentiometer is used for **(VCC (+ve))** and **(GND (-ve)) respectively**
    - Provide **supply** to the potentiometer from the breadboard
    - Connect the potentiometer's **output pin to an ESP32's analog input pin 36**. The ESP32's analog **input pin converts the voltage** (between 0v and 3.3V) **into**

integer values (between 0 and 4095), **or analog value**.



3. To Map value of voltage with potentiometer use **float map() function**. It will map 0 to 3.3V/5V according to the rotation of the **potentiometer**.It will map value from low to high or high to low.
   - **floatMap()** is used to get all decimal value

```
float floatMap(float x, float in_min, float in_max, float out_min, float out_max) {
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

4. Initialize using **void setup()** function
   - **Serial. begin(9600)** is used for data exchange speed. This tells the Arduino to get ready to exchange messages with the Serial Monitor at a data rate of 9600 bits per second. That's 9600 binary ones or zeros per second and is commonly called **a baud rate**.

```
void setup() {
  Serial.begin(9600);
}
```

5. To execute main process write the **void loop()**

- The output pin of the potentiometer is connected to the ESP32 analog input pin. To read this value **use analogRead()** function.
  - **analogRead()** function is used to read the value of the potentiometer reading
- Rescale to the potentiometer's angle by using **float map()** function.
- Rescale to the potentiometer's voltage. By using the following:
  - **Serial.print()** is used to print the value, print Analog, and then analog value, voltage
  - **Serial.println**(volage*1000) is used to show voltage in terms of 1.00

```
void loop() {

  int analogValue = analogRead(4);

  float voltage = floatMap(analogValue, 0, 4095, 0, 3.3);

  Serial.print("Analog: ");
  Serial.print(analogValue);
  Serial.print(", Voltage: ");
  Serial.println(voltage*1000);

  delay(1000);
}
```

6. Compile and upload the program to **ESP32 board** using Arduino IDE
   - Verify the program on clicking **Tick** option
   - Upload the program on clicking **arrow** option
     - If port is not selected, insert the USB cable in Computer's port and select the port
   - Go to **Tools** and select **Serial Monitor**
     - Rotate the potentiometer and see the output

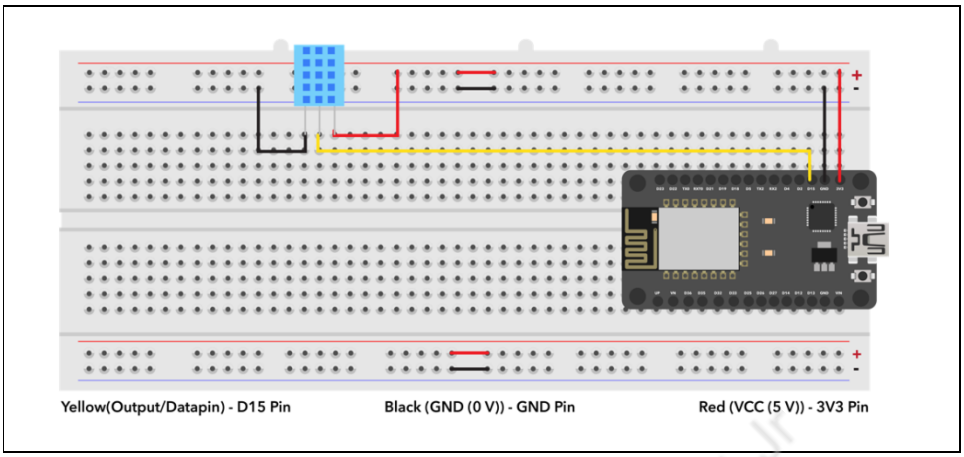
- By rotating the potentiometer knob, observe the voltage value. The voltage will rise or fall.

7. To read the temperature and humidity value from the DHT11 sensor and print it to Serial Monitor and check the same on the graph

   - Gather the material from the IoT kit:
     - 1 x ESP32
     - 1 x USB Cable
     - 1 x Breadboard
     - 4 x Jumper wires
     - 1 x Potentiometer
     - 1 x Rotary Potentiometer

8. Connections for **Circuit** Diagram

   - Supply positive(VCC (+ve)) from the ESP 32 to breadboard terminal
   - Supply negative(GND (-ve)) from the ESP 32 to breadboard negative terminal
   - Take the DHT11 sensor, female jumper wires are already connected with DHT11
   - Take three male jumper wires insert into DHT11 sensor
   - connect VCC (+ve) of DHT11 with VCC (+ve) of the breadboard
   - connect GND(-ve) of DHT11 with GND(-ve) of the breadboard
   - Connect data/output pin of DHT11 with D15 of the ESP32



9. Write the code:
   - Define Pins
     - define DHTPIN 15
     - define DHTPIN DHT11

```
#define DHTPIN 15

#define DHTTYPE DHT11


DHT dht(DHTPIN, DHTTYPE);
```

   - Initialize the setup()
     - Serial. begin(9600) is used for data exchange speed.. This tells the Arduino to get ready to exchange messages with the Serial Monitor at a data rate of 9600 bits per second. That's 9600 binary ones or zeros per second and is commonly called a baud rate.
     - Serial.print used to print data.
     - dht.begin() is used to begin the process

```
void setup() {
  Serial.begin(9600);
  Serial.println("DHT11 sensor!");
  //call begin to start sensor
  dht.begin();
}
```

   - **To execute the main process write the void loop()**
     - Create float h and float t variable to store decimal value

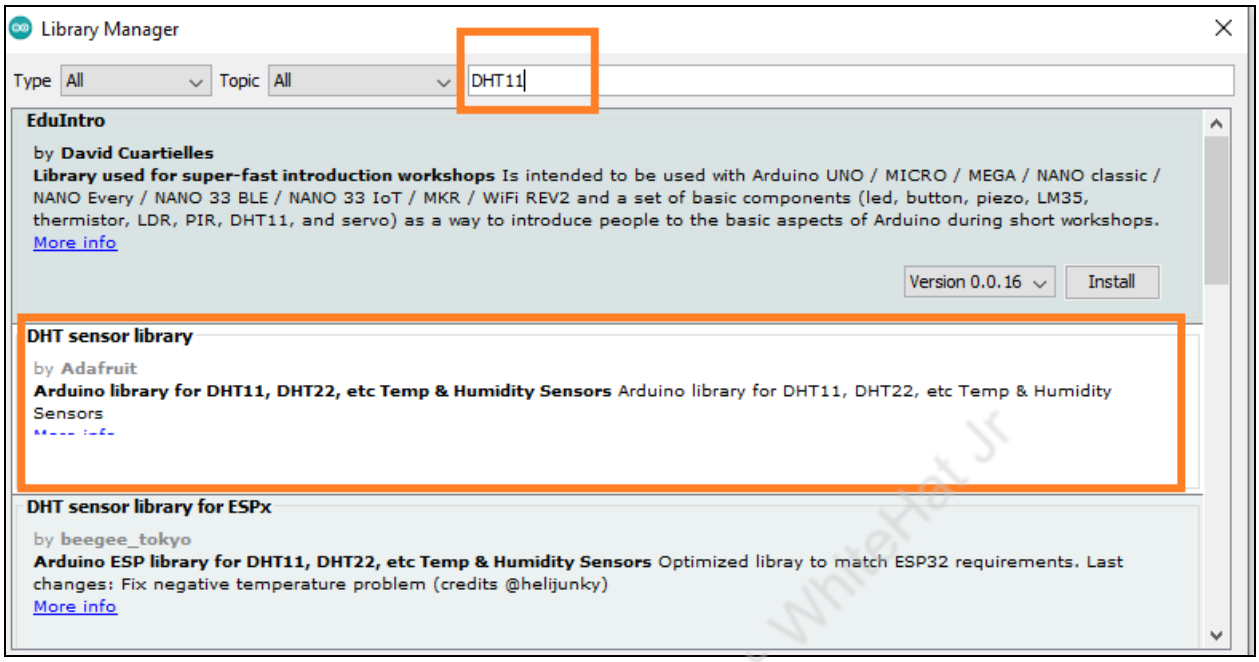
- **readHumidity()** will read the sensor's humidity value.
- **readTemperature()** will read the sensor's temperature value.
- Check if any reads failed and exit early using isnan()
- **Serial.println** used to print data. Print ("Failed to read from DHT sensor!")
- Return the process using **return()**
- **Serial.print()** is used to print the value, print Humidity, (h) , , Temperature, (t)
- Set **delay** of 2000ms

```
void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  // print the result to Terminal
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(",");
  Serial.print("Temperature: ");
  Serial.println(t);
  delay(2000);
}
```

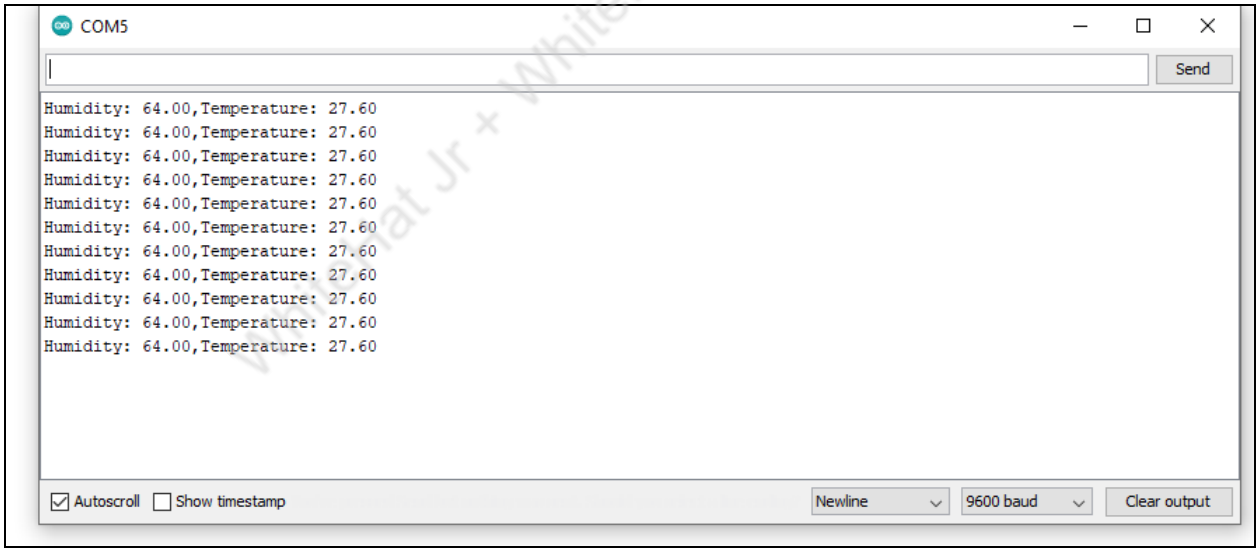
- Compile and upload the program to ESP32 board using Arduino IDE
  - Verify the program on clicking Tick option
  - Upload the program on clicking arrow option
- If port is not selected, insert the USB cable in Computer's port and select the port
  - Go to Tools and select Serial Monitor
  - Rotate the potentiometer and see the output
- If error message comes like no such file or directory then use the below method to resolve this error

```
DHT.h: No such file or directory
compilation terminated.
exit status 1
DHT.h: No such file or directory
```

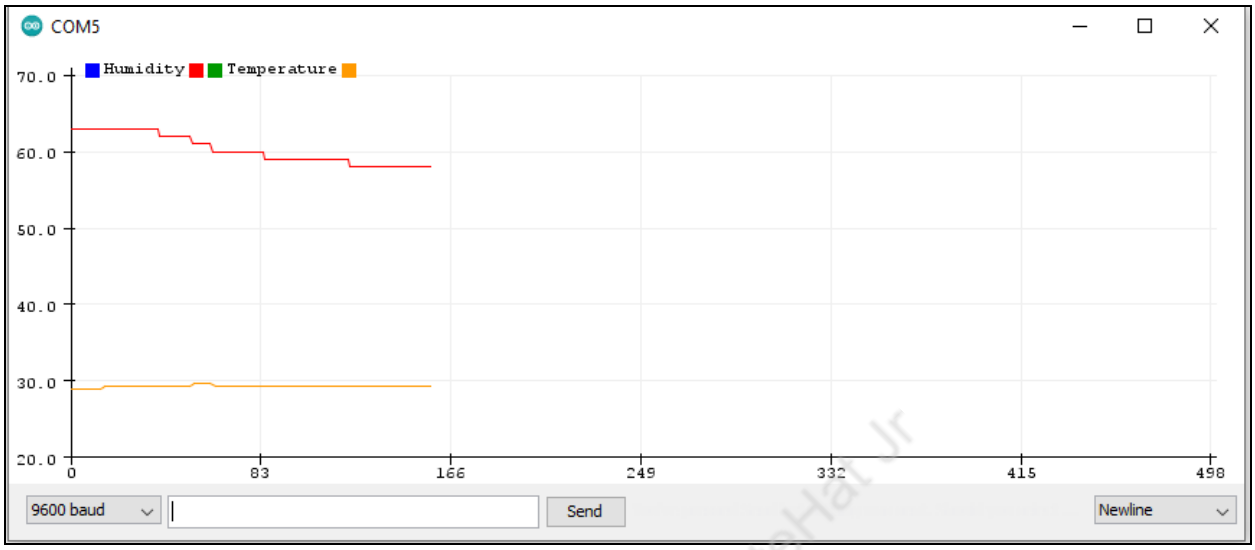
- Go to Tools
  - Click on Manage Libraries
  - Write the component name which need to install
  - Click on Install



- Go to Tools and select Serial Monitor
  - See the Humidity and Temperature value



- To verify the analog waveform
  - Go to Tools and select Serial Plotter
  - See the Humidity and Temperature value



- Observe the reading of humidity and temperature.

What's NEXT?

In the next class, we will learn about RGB Led

Expand Your Knowledge

To know more about **Potentiometer** click here.

WhiteHat Jr + WhiteHat Jr + WhiteHat Jr