Blynk IOT Temperature Sensor

ESP8266- Wifi Board

Muhammad Arslan Babur

Instrumentation System

An Instrumentation system is a device that "converts an unknown quantity into a record or display which human faculties can interpret." [13]. This unknown quantity is often an external stimulus that influences the changes in properties of an entity that are measurable. Hence by relating the known change of property by the change in stimuli; it can be quantified and thus monitored and controlled. In this project, the stimulus to be measured is temperature - a fundamental scalar quantity - using a thermistor device.

The following figure will illustrate and visualize the thermistor's instrumentation system via a block diagram. After which each block/step will be explained.

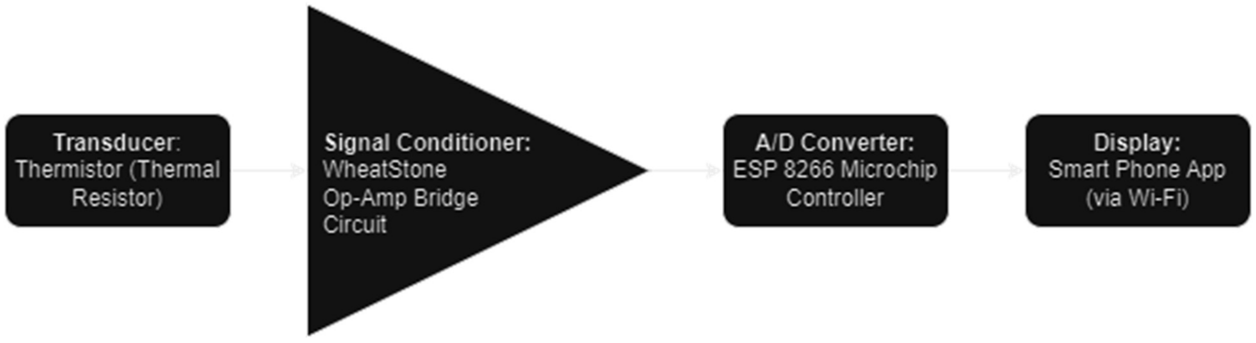

Transducer: Thermistor (Thermal Resistor)
Signal Conditioner: WheatStone Op-Amp Bridge Circuit
A/D Converter: ESP 8266 Microchip Controller
Display: Smart Phone App (via Wi-Fi)

Figure. Block Diagram of Thermistor Instrumentation System

Transducer:

The diagram starts with the input transducer; the sensor that first reacts to the external stimuli - temperature; which is the thermal resistor or Thermistor. This type of resistive transducer experiences a decrease in resistance as the temperature that is sensed by the component surface increases. The change in resistance to temperature is not always linear, which was proven to be the case in this project.

Signal Conditioner:

A signal conditioner is often a circuit that is used to prepare the transducer output signal - in this case change in resistance - to be read by the computer/microchip controller. The signal conditioner used in this temperature sensor is a Wheatstone bridge circuit. This circuit consists of 4 resistors in a diamond or “bridge” configuration; three of which act as reference and the fourth being the thermistor. A voltage divider can then be connected (like a bridge) between the reference and the thermistor sides; and the potential difference can be obtained and input into the Op-Amp Circuit featuring a microchip.
The signal can then be said to be conditioned after the Op-Amp circuit outputs an analog signal in the form of voltage, which is to be sent to the A/D converter and processor.

Please refer to the next chapter for a more detailed explanation of the signal conditioning circuit used in this project.

A/D Converter & Computer:

This component in the instrumentation system is normally solely for the purpose of converting the analog voltage signal to a digital one that can be understood by the computer. However, the computer used in this project is equipped with an in-built A/D converter pin; hence reducing the number of components needed.

The microchip controller used in this project is the ESP 8266 board. One of the most common boards following the Arduino Uno. The decision to use the

ESP board was due to its built-in Wi-Fi transmitter, which will be used to send the digital output to the display - a smartphone. This feature was found lacking in the Arduino Uno version that was available.

The ESP computer chip was programmed to perform a series of calculations in order to convert the digital output to the final temperature reading and send the output to, via Wi-Fi, to the display.

Please refer to the next chapter for a more detailed explanation on the code used to program the ESP 8266 board.

Display

The display is the final station to obtain the output of the thermistor. instrumentation system. It receives the data transmitted from the ESP board via Wi-Fi and "prints" it to the smartphone screen via an application.

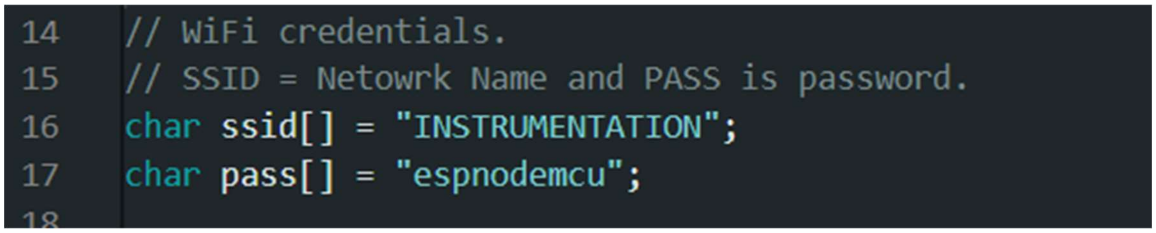
ESP Programming

Aside from mechanical and electrical design of the instrumentation system, the pivotal factor in its operation was the programming for the esp8266 WiFi board within the system. The code allowed the circuit to broadcast the sensor data obtained wirelessly to an app that could be monitored easily by the user. For this project, the programming was done in Arduino IDE, with the following libraries included.

```
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
```

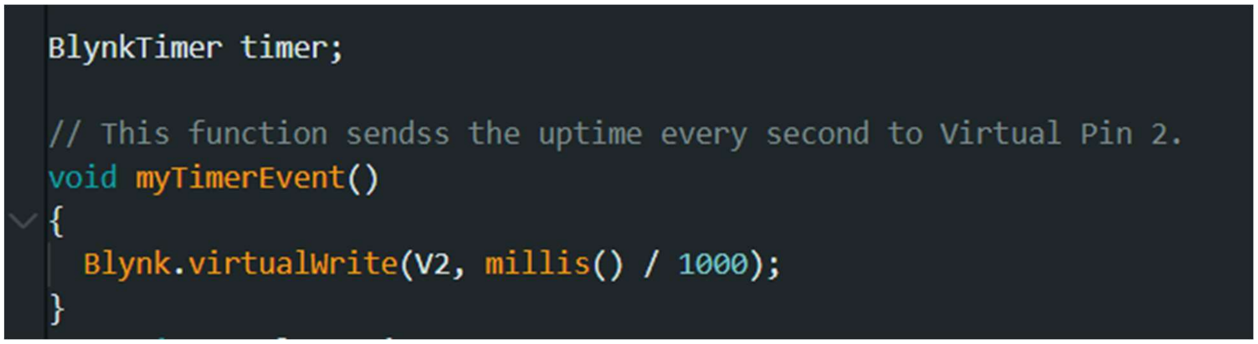
The ESP8266WiFi.h allows the microchip board to connect to a mobile hotspot to gain wireless capabilities, whereas the BlynkSimpleEsp8266.h allows the transmission of data to Blynk Cloud which can then be viewed on the app. The main principles behind the program are very basic.

- Credentials for the mobile hotspot are provided to the microprocessor



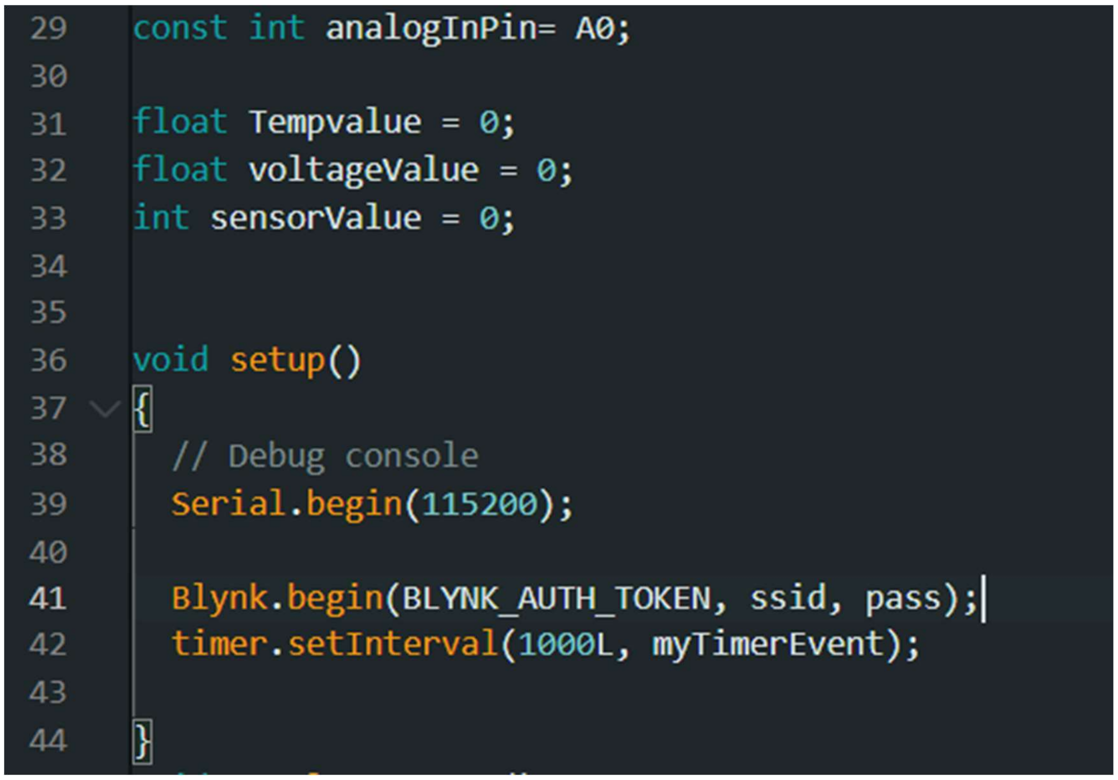
```
// WiFi credentials.
// SSID = Netowrk Name and PASS is password.
char ssid[] = "INSTRUMENTATION";
char pass[] = "espnodemcu";
```

- A timer event with uptime is set for the Blynk Cloud to monitor the operation time of the board.



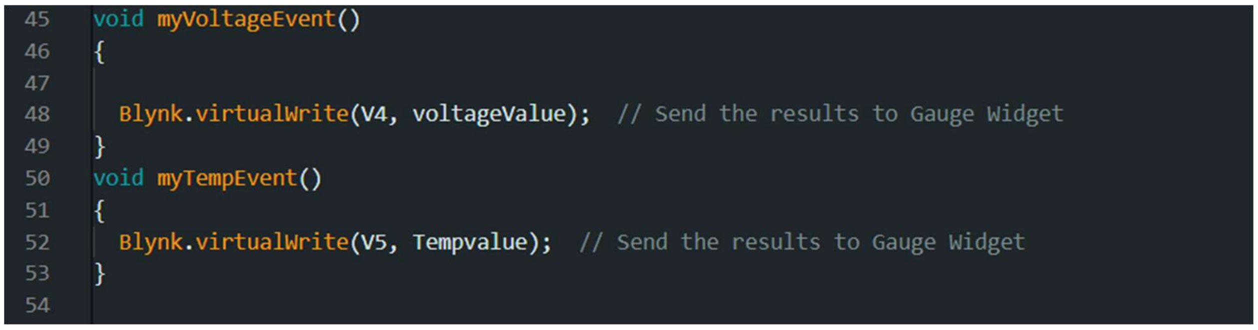
```
BlynkTimer timer;

// This function sendss the uptime every second to Virtual Pin 2.
void myTimerEvent()
{
  Blynk.virtualWrite(V2, millis() / 1000);
}
```

- The constants to be measured are defined, along with a setup command to configure the esp8266 board with the Blynk application account using an authorization token and password.



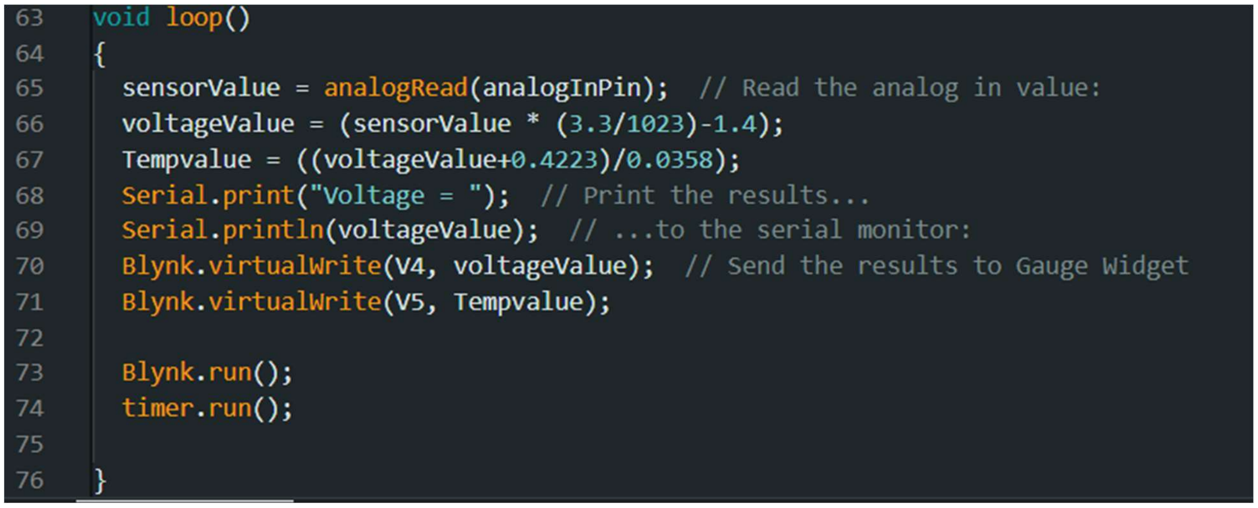
```
const int analogInPin= A0;

float Tempvalue = 0;
float voltageValue = 0;
int sensorValue = 0;


void setup()
{
  // Debug console
  Serial.begin(115200);

  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  timer.setInterval(1000L, myTimerEvent);

}
```

- Create the events for voltage and temperature values to be virtually written to Blynk data streams.



```
void myVoltageEvent()
{

  Blynk.virtualWrite(V4, voltageValue);  // Send the results to Gauge Widget
}
void myTempEvent()
{
  Blynk.virtualWrite(V5, Tempvalue);  // Send the results to Gauge Widget
}

```

- Lastly, a void loop is set up to ensure that the circuit measures the value from the probe with the timing intervals set to initiate the Timer and Blynk run commands.



```
void loop()
{
  sensorValue = analogRead(analogInPin);  // Read the analog in value:
  voltageValue = (sensorValue * (3.3/1023)-1.4);
  Tempvalue = ((voltageValue+0.4223)/0.0358);
  Serial.print("Voltage = ");  // Print the results...
  Serial.println(voltageValue);  // ...to the serial monitor:
  Blynk.virtualWrite(V4, voltageValue);  // Send the results to Gauge Widget
  Blynk.virtualWrite(V5, Tempvalue);

  Blynk.run();
  timer.run();

}
```

In summary, the code ensures that the circuit outputs and receives the data as needed for the system to function properly. The data is processed using the code, whereas the electrical systems ensure the data is measured properly. The sync with the Blynk account is then used to display the results of the sensor and code towards the end user.