Collecting and Analyzing Data with Raspberry Pi

Introduction and Objective

The students will perform practical work by programming a Raspberry Pi to collect and analyze temperature data. The teacher will guide them in using Python for data processing.

Objective

To learn how to use a Raspberry Pi computer to read temperature from a sensor and use the Python programming language to make sense of that data.

1. Why Raspberry Pi?

Unlike the smaller microcontrollers (like Arduino) we used before, the Raspberry Pi is a full computer.

- It has an OS: It runs an operating system (Linux) just like a laptop.
- It runs Python: We can write powerful scripts directly on the device.
- It stores data: We can save temperature readings to files to look at later.

2. Connecting the Sensor

We use the GPIO pins (General Purpose Input/Output) on the Raspberry Pi to connect our sensor.

- Pin 1 (3.3V Power): Provides electricity to the sensor.
- Pin 6 (Ground): Completes the electrical circuit.
- GPIO Pin (e.g., GPIO 4): Receives the data signal from the sensor.

->The Tools We Need

- Raspberry Pi: The computer.
- Thonny IDE: A simple program pre-installed on the Raspberry Pi for writing Python code.
- Libraries: Pre-written code packages (like Adafruit_DHT) that help Python talk to the sensor.

STEPS:

Step 1: Import Libraries

First, we tell Python to use the special tools for our sensor.

- **import Adafruit_DHT** (Loads the sensor tool)
- **import time** (Allows us to pause between readings)

Step 2: Reading the Data

We write a loop (a repeating instruction) to check the sensor every few seconds.

- The command **sensor.read()** asks the sensor: "What is the temperature now?"
- If the sensor answers, we see two numbers: Humidity and Temperature.

Step 3: Printing the Output

We use the print() command to show the results on the screen:

```
Example Output: "Temp: 25.0 C | Humidity: 60%"
```

Storing Data in a List

To analyze data, we can't just print it; we must remember it. We use a Python List.

```
temps = [25, 26, 24, 25, 27]
```

Basic Analysis

Once we have a list of temperatures, we can ask Python questions about it:

- Maximum: What was the hottest point? (max(temps))
- Minimum: What was the coldest point? (min(temps))
- Average: What is the general temperature? (Add all numbers and divide by the count).

Why do we do this? This turns raw numbers into useful information. For example, knowing the average temperature helps us decide if a room is generally too hot or too cold.

Conclusion

By combining the Raspberry Pi with Python, we can do more than just see the temperature; we can record it, analyze it, and make decisions based on the data (like turning on a fan if the average gets too high).

Review Questions

- 1. What is the name of the coding program we use on Raspberry Pi? We typically use Thonny IDE to write our Python scripts.
- 2. Why do we import "libraries" in Python? Libraries give us pre-written code so we don't have to write complex instructions from scratch to talk to the sensor.
- 3. If we want to find the hottest temperature recorded, which Python function do we use? We use the max() function on our list of data.