MULTIPLE SENSOR DATA COLLECTION AND TRANSMISSION

May 20, 2023

By Zameel Ali Mohammed

OVERVIEW

1. Introduction

The Sensor Data Collection and Transmission project aims to collect data from various sensors and transmit it to the ThingSpeak platform. The project utilizes the Bolt IoT device along with temperature, moisture, and Light sensor (LDR) to gather real-time environmental data. The collected data is then converted into meaningful values and transmitted to ThingSpeak for storage, analysis and visualization.

2. Methodology

The project follows a systematic approach to collect sensor data and transmit it to ThingSpeak. It involves setting up the hardware components, writing the necessary software code, and implementing data collection and transmission functions.

3. Hardware Setup

The hardware setup consists of the following components

- Bolt IoT device: Acts as the central hub for data collection and transmission.
- Temperature sensor: Measures the ambient temperature.
- Moisture sensor: Measures the soil moisture level.
- Light sensor: Measures the ambient light intensity.

4. Software Implementation

The project utilizes the bolt Python library to interact with the Bolt IoT device and sensors. The software code is written in Python, which provides a flexible and easy-to-use programming language for IoT applications. The code is organized into functions to perform specific tasks such as selecting a sensor, reading sensor data, and transmitting data ta ThingSpeak.

5. Software Code - Python

Here's a detailed explanation of each line of the program:

```
from boldest import Bolt

import time

import requests

import json
```

- *The program imports necessary modules: '**Bolt**' from the '**boltiot**' package, '**time**', 'requests' and '**json**'. These modules provide functionality for interacting with the Bolt device, manage time delay, making HTTP requests and handling JSON data.*

```
# Bolt Cloud API Configuration

api_key = "YOUR_API_KEY"

device_id = "YOUR_DEVICE_ID"

bolt = Bolt (api_key, device_id)
```

- *API key and device ID for the Bolt Cloud API are assigned to variables '**api_key**' and '**device_id**', respectively.*
- *An instance of the '**Bolt**' class is created by passing the API key and device ID as arguments. This instance, named '**bolt**', will be used to interact with the Bolt device.*

```
# ThingSpeak Configuration

THINGSPEAK_API_KEY = "YOUR_THINGSPEAK_API_KEY"

THINGSPEAK_URL = f"https://api.thingspeak.com/update?api_key={THINGSPEAK_API_KEY}"
```

- *ThingSpeak API key is assigned to the variable '**THINGSPEAK_API_KEY**'.*
- *The ThingSpeak URL for updating channel data is constructed by combining the base URL with the API key.*

```
# Analog Pin

analog_pin = "A0"
```

- *The analog pin to read sensor data from is assigned to the variable '**alanog_pin**'. In this case, it is set to **"A0"** because we have only one analog pin on Bolt device.*

```
# Multiplexing Digital Pins

pins = {

  "Temp": "0",

  "Mos": "1",

  "Lig": "2"

}
```

- *A dictionary named '**pins**' is defined to map sensors type to their corresponding digital pins on the Bolt device.*
- *In this case, the "**Temp**" sensor is mapped to "**0**", "**Mos**" sensor is mapped to "**1**", and "**Lig**" sensor is mapped to "**2**".*

def select_sensor_type(sensor_type):

"""

Selects a specific sensor by setting the corresponding digital pin to HIGH.

Args:

sensor_type (str): The type of sensor to select.

"""

response = bolt.digitalWrite(pins[sensor_type], 'HIGH')

print(f"{sensor_type} is selected and the response is {response}")

time.sleep(60)

- *A function named '**select_sensor_type**' is defined to select a specific sensor by setting the corresponding digital pin to '**HIGH**'.*
- *It takes '**sensor_type**' as an argument, which represents the type of sensor to select.*
- *The '**digitalWrite**' method is stored in the 'response' variable.*
- *A message is printed to the console indicating which sensor is selected and the response received.*
- *A time delay of 60 seconds is added to ensure a 1-minute delay before proceeding.*

def deselect_sensors(sensor_type):

"""

Deselects a specific sensor by setting the corresponding digital pin to LOW.

Args:

sensor_type (str): The type of sensor to deselect.

"""

response = bolt.digitalWrite(pins[sensor_type], 'LOW')

print(f"{sensor_type} is deselected and the response is {response}")

time.sleep(60)

- *A function named '**deselect_sensor**' is defined to select a specific sensor by setting the corresponding digital pin to LOW.*
- *It takes '**sensor_type**' as an argument, which represents the type of sensor to select.*
- *The '**digitalWrite**' method of the '**bolt**' object is called with the pin value obtained from the '**pin**' dictionary and the state set to '**LOW**'.*
- *The '**digitalWrite**' method is stored in the '**response**' variable.*
- *A message is printed to the console indicating which sensor is selected and the response received.*
- *A time delay of 60 seconds is added to ensure a 1-minute delay before proceeding*

def get_sensor_data(pin, sensor_type):

"""

Reads sensor data from the specified pin.

Args:

```
    pin (str): The pin to read data from.
    sensor_type (str): The type of sensor being read.
  Returns:
    int: The sensor value read from the pin.
  """
  select_sensor_type(sensor_type)
  response = bolt.analogRead(pin)
  print(f"Reading {sensor_type} data and response is {response}")
  deselect_sensors(sensor_type)
  data = json.loads(response)
  sensor_value = int(data["value"])
  return sensor_value
```

- *A function named '**get_sensor_data**' is defined to read sensor data from the specified pin.*
- *It takes '**pin**' and '**sensor_type**' as arguments, representing the pin to read data from and the type of sensor being read, respectively.*
- *The '**select_sensor_type**' function is called to select the specified sensor.*
- *The '**analogRead**' method of the 'bolt' object is called with the 'pin' arguments to read the sensor data.*
- *The response from the '**analogRead**' method is stored in the 'response' variable.*
- *A message is printed to the console indicating which sensor data is being read and the response received.*
- *The '**deselect_sensors**' function is called to deselect the specified sensor.*
- *The response data is parsed as JSON using the '**json.loads**' function, and the sensor value is extracted from the parsed data.*
- *The sensor value is converted to an integer and returned.*

```
def send_to_thingspeak(data):
  """
  Sends sensor data to ThingSpeak.
  Args:
    data (dict): The sensor data to send.
  """
  payload = {
    "api_key": THINGSPEAK_API_KEY,
    "field1": data["temperature"],
    "field2": data["moisture"],
    "field3": data["light"]
  }
```

```
    requests.get(THINGSPEAK_URL, params=payload)
    print("Data sent to ThingSpeak")
```

- *A function named '**send_to_tingspeak**' is defined to send data to the ThingSpeak platform.*
- *It takes '**data**' as an argument, which is a dictionary containing the sensor data.*
- *The data is formatted as a payload dictionary with the ThingSpeak API key add the field values for temperature, moisture, and light.*
- *A GET request is made to the ThingSpeak URL with the payload as parameters to send the data.*
- *A message is printed to console indicating that the data has been sent to ThingSpeak.*

```
def collect_and_transmit_data():
    """
    Collects sensor data and transmits it to ThingSpeak.
    """
    print("Collecting Sensors Data")
    temperature = get_sensor_data(analog_pin, "Temp")
    moisture = get_sensor_data(analog_pin, "Mos")
    light = get_sensor_data(analog_pin, "Lig")

    # Converting RAW data
    temp_adc_val = temperature * 4.88
    temp_val = temp_adc_val / 10
    moisture_percentage = 100 - (moisture / 1023.0) * 100

    # Prepare data in the desired format
    sensor_data = {
        "temperature": temp_val,
        "moisture": moisture_percentage,
        "light": light
    }

    # Send data to ThingSpeak
    send_to_thingspeak(sensor_data)
```

- *A function named '**collect_and_transmit_data**' is defined to collect sensor data and transmit it to ThingSpeak.*
- *A message is printed to the console indicating that the sensor data collection is in process.*
- *The '**get_sensor_data**' function is called three times to obtain temperature, moisture, and light sensor readings from the '**analog_pin**'.*

- *The raw sensor data is converted to their respective real-world values. Temperature is multiplied by 4.88 and divided by 10, while moisture percentage is calculated using a formula.*
- *The converted data is stored in a dictionary named '**sensor_data**'.*
- *The '**send_to_thingspeak**' function is called with '**sensor_data**' as an argument to transmit the data to ThingSpeak.*

```
# Main program loop

while True:

    try:

        collect_and_transmit_data()

    except KeyboardInterrupt:

        print("Program terminated by user.")

        break

    except Exception as e:

        print("An error occurred:", str(e))
```

- *The main program loop is initiated using a '**while True**' statement.*
- *Inside the loop, the '**collect_and_transmit_data**' function is called to collect and transmit sensor data to ThingSpeak.*
- *The program handles two exceptions: '**KeyboardInterrupt**' (when the user terminates the program) and 'Exception' (for any other error).*
- *If a '**KeyboardInterrupt**' exception occurs, a message is printed, and the loop is broken to exit the program.*
- *If any other exception occurs, an error message is printed along with the exception details.*

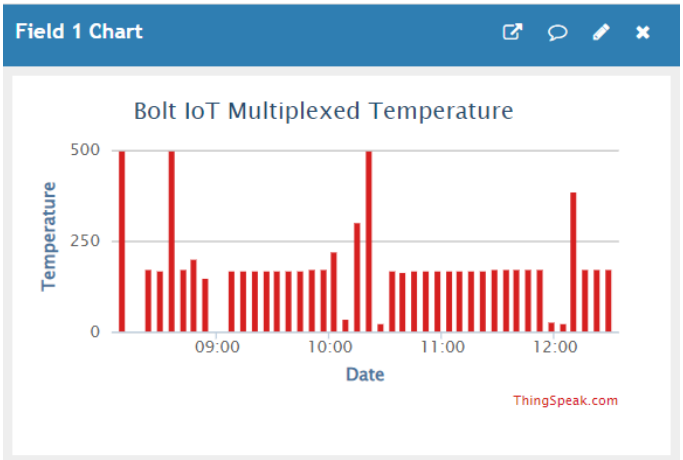
6. Data Collection and Conversion

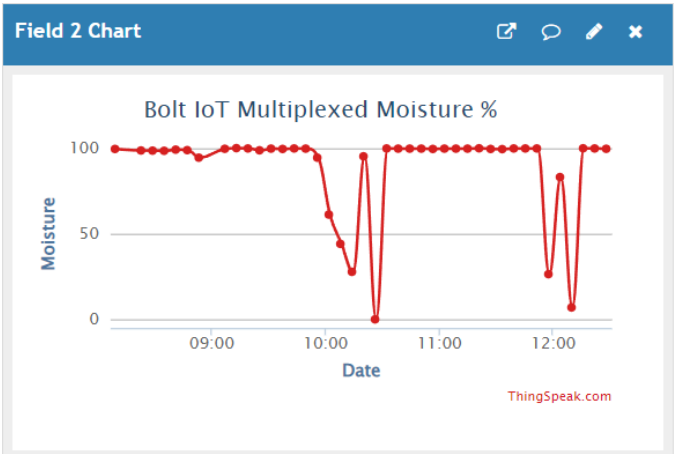
The data collection process involves reading analog sensor data from the Bolt IoT device.

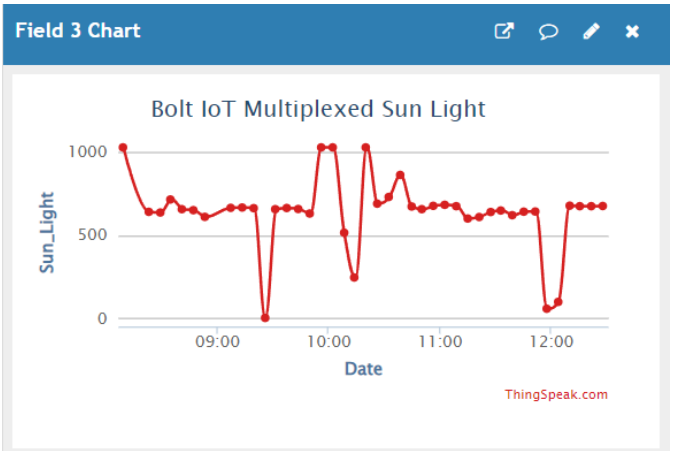
*The **select_sensor_type()** function is used to choose a specific sensor by setting the corresponding digital pin to HIGH. The **get_sensor_data()** function then reads the analog sensor value from the specified pin. The raw sensor data is converted into real-world values using appropriate formulas and calculations. For example, the temperature value is converted from ADC value to Celsius using the formula: temperature = (ADC*4.88)/10.*

7. Data Transmission to ThingSpeak

*Once the sensor data is collected and converted, it is transmitted to the ThingSpeak platform for storage and analysis. The **send_to_thingspeak()** function sends the sensor data as parameters in the payload to the ThingSpeak API. The payload includes the API key and field values corresponding to the temperature, moisture, and light readings. The requests library is used to send an HTTP GET request to the ThingSpeak API endpoint with the payload data.*







ThingSpeak channel sent data

8. Results and Observations

The project successfully collects sensors data from the temperature, moisture, and light sensor. The data is converted into real-world values, and the result are transmitted to ThingSpeak. By analyzing the data stored in ThingSpeak, various insights can be gained, such as temperature trends, soil moisture levels, and light intensity variations.

9. Conclusion

The Sensor Data Collection and Transmission project demonstrates the capability of the Bolt IoT device to collect data from multiple sensors and transmit it to the ThingSpeak platform. The project provides a foundation for building more advanced IoT applications that involve sensor data analysis and decision-making based on real-time environmental parameters.

Future enhancements to the project could include additional sensors, data visualization, and integration with other IoT platforms.

10. Reference

- Bolt IoT: Getting started with Bolt IoT
- ThingSpeak: IoT Analytics - ThingSpeak Internet of Things
- Python documentation: The Python Standard Library — Python 3.11.3 documentation
- Sensor datasheets: 1. Guide for LM35, LM335 and LM34 Temperature Sensors with Arduino
  2. Soil Moisture Sensor | How it's Works » Electro Duino
  3. What Is LDR? | Working Principle, Types, Applications, Projects

11. Appendix

- Source code: code
- Circuit diagram: image