Date: 14-06-2023

Greenhouse Monitoring with microcontrollers

Objective:

The aim of this project is to implement a client-server communication system for monitoring and controlling IoT devices. The server receives data from multiple clients, processes the data, and sends control actions back to the clients based on predefined rules. Client 1 monitors light intensity and controls an LED, while Client 2 monitors temperature and controls a buzzer.

Abstract:

This report presents the implementation and evaluation of a client-server communication system for IoT monitoring and control. The system consists of two clients equipped with sensors to monitor light intensity and temperature, respectively. The clients communicate with a server that processes the data and sends control actions back to the clients. The report provides an overview of the system, describes the methodology, presents the implementation details, reports the obtained results, and concludes with key findings and potential areas for further improvement.

System Overview & Methodology:

The client-server communication system follows a distributed architecture, where clients and the server communicate over a network. The clients are Raspberry Pi devices equipped with sensors, while the server runs on a host machine. The methodology involves the following steps:

1. Hardware setup: Configure the Raspberry Pi devices with the required sensors and connect them to the GPIO pins.
2. Software setup: Install the necessary libraries and dependencies on the Raspberry Pi devices and the host machine.
3. Client implementation: Develop separate code for each client to read sensor data, send it to the server, and receive control actions.
4. Server implementation: Create a server code to accept connections from the clients, receive data, process it based on predefined rules, and send control actions.
5. GUI interface: Develop a graphical user interface using the wxPython library to display real-time data and control actions for each client.

Implementation:

Server Setup:

- The server is implemented using Python and the socket library.
- The server creates a socket object, binds it to a specific IP address and port, and listens for incoming connections.
- A GUI interface is created using the wxPython library to display real-time data and actions for each client.
- A separate thread is started to handle each client connection, allowing concurrent communication.

Client 1 (Light Intensity Monitoring and Control):

- Client 1 runs on a Raspberry Pi and is responsible for monitoring light intensity and controlling an LED.
- Light intensity is measured using an LDR (Light Dependent Resistor) connected to a GPIO pin.
- The client establishes a socket connection with the server and continuously sends light intensity data to the server.

- The client receives control actions from the server and turns on or off the LED based on the received action.

Client 2 (Temperature Monitoring and Control):

- Client 2 also runs on a Raspberry Pi and monitors temperature using a DHT11 sensor connected to a GPIO pin.
- The client establishes a socket connection with the server and sends temperature data to the server at regular intervals.
- Control actions received from the server are used to control a buzzer connected to a GPIO pin on the Raspberry Pi.

Server Processing:

- The server receives data from each client, processes the data based on the client's IP address, and determines the appropriate control action.
- For Client 1, the server checks the light intensity and sends an action to turn on the LED if the intensity is below a certain threshold.
- For Client 2, the server checks the temperature and sends an action to turn on the buzzer if the temperature exceeds a predefined limit.

Results:

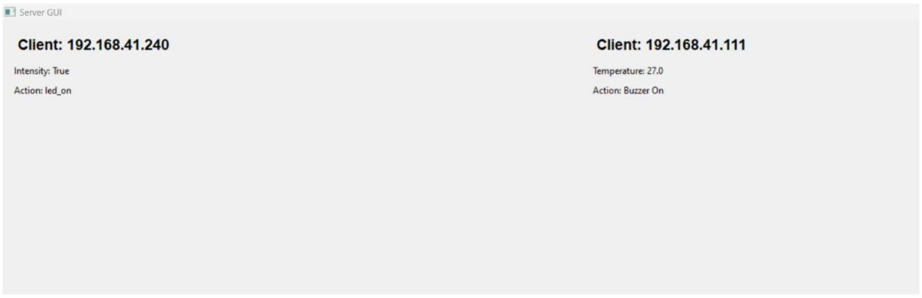

Fig (1): Graphical user interface using wx in server

```
2023-06-12 16:41:11,618 - Received data from 192.168.41.240: {'intensity': 0}
2023-06-12 16:41:11,633 - Sent action to 192.168.41.240: led_on
2023-06-12 16:41:13,443 - Received data from 192.168.41.111: 27.0
2023-06-12 16:41:13,443 - Sent action to 192.168.41.111: Buzzer On
2023-06-12 16:41:36,788 - Received data from 192.168.41.240: {'intensity': 0}
2023-06-12 16:41:36,846 - Sent action to 192.168.41.240: led_on
2023-06-12 16:41:54,024 - Received data from 192.168.41.111: 27.0
2023-06-12 16:41:54,032 - Sent action to 192.168.41.111: Buzzer On
2023-06-12 16:42:01,893 - Received data from 192.168.41.240: {'intensity': 0}
2023-06-12 16:42:01,934 - Sent action to 192.168.41.240: led_on
2023-06-12 16:42:27,124 - Received data from 192.168.41.240: {'intensity': 0}
2023-06-12 16:42:27,140 - Sent action to 192.168.41.240: led_on
2023-06-12 16:42:37,202 - Received data from 192.168.41.111: 27.0
2023-06-12 16:42:37,202 - Sent action to 192.168.41.111: Buzzer On
2023-06-12 16:42:52,169 - Received data from 192.168.41.240: {'intensity': 0}
2023-06-12 16:42:52,185 - Sent action to 192.168.41.240: led_on
2023-06-12 16:43:17,244 - Received data from 192.168.41.240: {'intensity': 1}
2023-06-12 16:43:17,254 - Sent action to 192.168.41.240: led_on
2023-06-12 16:43:17,855 - Received data from 192.168.41.111: 27.0
2023-06-12 16:43:17,855 - Sent action to 192.168.41.111: Buzzer On
2023-06-12 16:44:00,966 - Received data from 192.168.41.111: 27.0
2023-06-12 16:44:00,966 - Sent action to 192.168.41.111: Buzzer On
```

Fig (2): Date base created in server using Logging

Fig (3): Outcome from client1 (Raspberry pi)

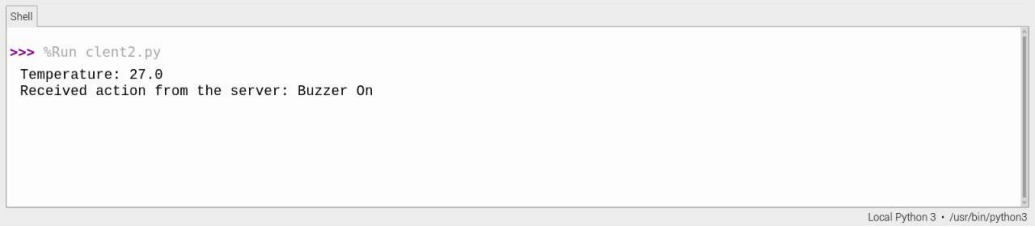

Fig (4): Outcome from client2 (Raspberry pi)

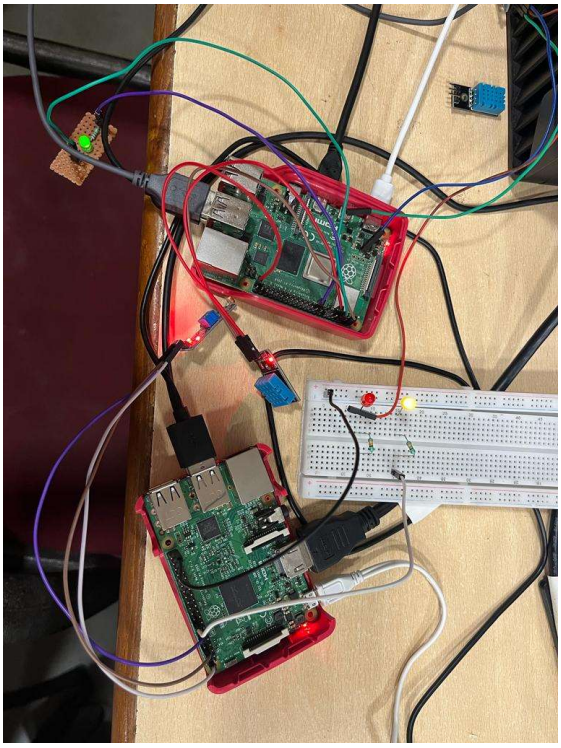

Fig (5): Hardware implementation

Conclusion:

The implementation of the client-server communication system demonstrated the feasibility of real-time monitoring and control of IoT devices. The system showcased how sensor data can be collected from multiple devices, processed centrally by a server, and control actions can be communicated back to the devices. The utilization of a distributed architecture facilitated scalability and flexibility, enabling the addition of more clients with different functionalities in the future. The system can be further enhanced by incorporating additional sensors, implementing more sophisticated control algorithms, and integrating advanced features such as data logging and remote access.

This project highlights the significance of client-server communication in IoT applications, enabling effective monitoring, control, and automation. It serves as a foundation for developing more complex IoT systems that can cater to various domains, such as home automation, industrial monitoring, and environmental sensing. By harnessing the power of networked devices and intelligent decision-making algorithms, IoT-based solutions can enhance efficiency, reduce manual intervention, and improve overall system performance. The client-server communication model lays the groundwork for building robust and scalable IoT ecosystems.

References:

1. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "A Survey on Sensor Networks," in IEEE Communications Magazine, vol. 40, no. 8, pp. 102-114, August 2002, doi: 10.1109/MCOM.2002.1024422.

2. J. Novotny, P. Pisa and T. Fujdiak, "IoT-based Greenhouse Monitoring and Control System," 2016 IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI), Herl'any, Slovakia, 2016, pp. 155-158, doi: 10.1109/SAMI.2016.7439687.

3. Y. Liang and W. Li, "Design and implementation of remote monitoring system based on Internet of Things," 2012 IEEE 14th International Conference on Communication Technology (ICCT), Chengdu, China, 2012, pp. 877-881, doi: 10.1109/ICCT.2012.6511163.

Names: H Kesava Sravan
G Kavya Sudha
L Moditha Chowdary
R Krishna Prasath

Roll Number: CB.EN.U4ELC20023
CB.EN.U4ELC20020
CB.EN.U4ELC20036
CB.EN.U4ELC20032

Component	Marks
Topic (2)	
Implementation & Results (5)	
Report (2)	
Total (9)	