LABORATORY REPORT

MECHATRONICS SYSTEM INTEGRATION

MCTA 3203

SEMESTER 2 2023/2024

WEEK 10

LAB: SYSTEMS INTEGRATION

SUBMISSION DATE:

GROUP: 6

	NAME	MATRIC NUMBER
1.	MUHAMMAD AZIM ZUFAYRI BIN NORAZAM	2211973
2.	MUHAMMAD AMMAR FARIS BIN ARAFAS	2212645
3.	AHMAD ADAM BIN NORHELME	2219973
4.	AHMAD HAFIZULLAH BIN IBRAHIM	2216185

TABLE OF CONTENTS

ABSTRACT

The goal of this project is to create a system that uses the Modbus communication protocol to integrate sensors, a microcontroller, and a computer. OpenPLC is an additional Modbus Slave in the system, Arduino is the Modbus Slave, and Raspberry Pi is the Modbus Master. The system's functions include reading sensor data, controlling outputs in response to sensor readings, and displaying real-time sensor data monitoring.

INTRODUCTION

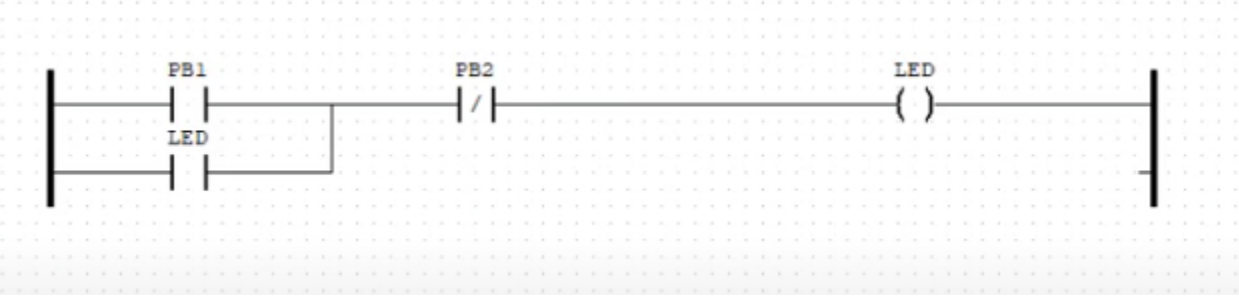
Effective data exchange in industrial automation and control systems is made possible by the integration of sensors, microcontrollers, and computers via the Modbus communication protocol. Devices can communicate with each other more easily when they are set up in a master-slave configuration, with the master initiating contact and managing data exchange with the slave devices.

MATERIALS AND EQUIPMENT

1. Raspberry Pi (Modbus Master)
2. Arduino Uno (Modbus Slave)
3. OpenPLC
4. Sensors Modbus

METHODOLOGY

1. Configure Raspberry Pi as the Modbus Master to communicate with Arduino for sensor data reading.
2. Set up Arduino as the Modbus Slave to read data from sensors.
3. Configure OpenPLC as a Modbus Slave to receive data from Raspberry Pi and control outputs based on sensor readings.
4. Implement Modbus communication protocols for seamless data exchange.
5. Demonstrate real-time monitoring of sensor data using suitable software.



Arduino

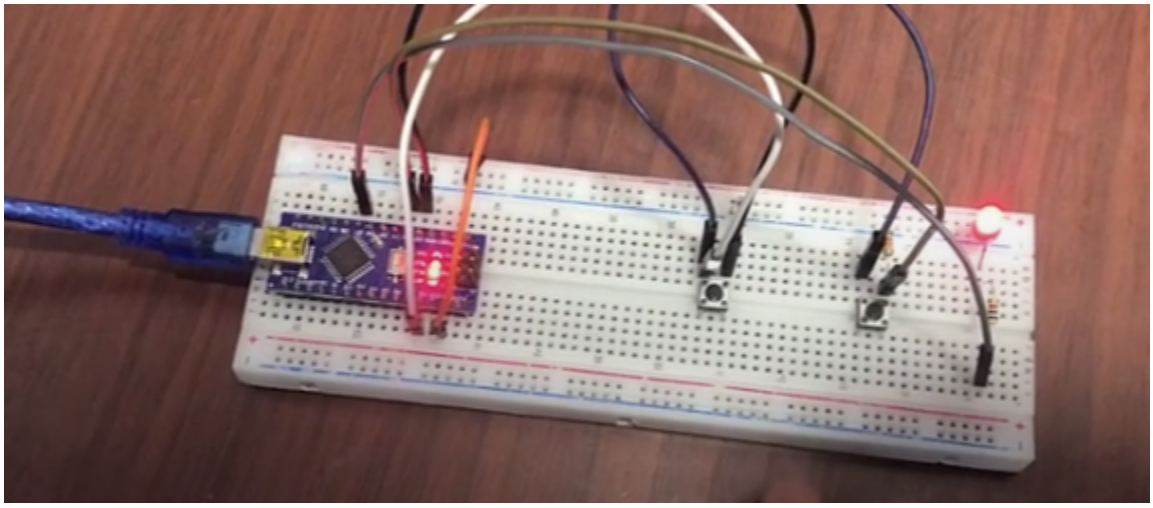
```
#include <ModbusRTU.h>

ModbusRTU mb;

const int BUTTON_PIN = 2;  // Pin where the button is connected

bool cbRead(Modbus::ResultCode event, uint16_t transactionId, void* data) {
  return true;
}

void setup() {
  Serial.begin(9600);
  pinMode(BUTTON_PIN, INPUT_PULLUP); // Button pin as input with internal pull-up

  mb.begin(&Serial, 9600);
  mb.slave(1); // Slave ID

  mb.addHreg(1000); // Register for button state
}

void loop() {
  mb.task();

  int buttonState = digitalRead(BUTTON_PIN);
  mb.Hreg(1000, buttonState);

  delay(100);
}
```

Raspberry Pi

```
from pymodbus.client.sync import ModbusSerialClient as ModbusClient
import RPi.GPIO as GPIO
import time

# Configure Modbus client
client = ModbusClient(method='rtu', port='/dev/ttyUSB0', baudrate=9600, timeout=1)
client.connect()

# Raspberry Pi LED pin configuration
LED_PIN = 17
```

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED_PIN, GPIO.OUT)

try:
    while True:
        # Read button state from Arduino (Slave ID 1)
        button_result = client.read_holding_registers(1000, 1, unit=1)

        if not button_result.isError():
            button_state = button_result.registers[0]
            print(f"Button State: {button_state}")

            # Control LED based on button state
            if button_state == 0:  # Assuming button is pressed when state is 0
                GPIO.output(LED_PIN, GPIO.HIGH)  # Turn LED on
            else:
                GPIO.output(LED_PIN, GPIO.LOW)   # Turn LED off

        time.sleep(0.1)
except KeyboardInterrupt:
    print("Program interrupted")
finally:
    GPIO.cleanup()
    client.close()
```

RESULTS



DISCUSSION

This system's use of the Modbus protocol allows for dependable and effective communication between the devices, facilitating smooth data interchange and control features. The system shows how Modbus can be used practically in circumstances involving industrial automation.

CONCLUSION

The project effectively demonstrates how to use the Modbus communication protocol to integrate sensors, microcontrollers, and computers. The system efficiently reads sensor data, allows real-time monitoring of sensor data, and regulates outputs based on sensor readings.

RECOMMENDATIONS

1. Optimise the system even more for increased scalability and performance.

2. Examine the Modbus protocol's further features and uses in industrial automation.

ACKNOWLEDGEMENT

A special thanks go out to Dr Wahju Sediono, Dr Ali Sophian, Dr Zulkifli Bin Zainal Abidin, our teaching assistants and our peers for their invaluable help and support in finishing this report. Their advice and guidance have greatly improved the quality and understanding of the project. Their commitments are greatly appreciated.

STUDENT DECLARATION

We, hereby declare that this project is entirely our work except for the documents that were given as references. Any external sources utilized for reference or inspiration have been properly cited and credited.

azim

Muhammad Azim Zufayri Bin Norazam

ammar

Muhammad Ammar Faris Bin Arafas

hafiz

Ahmad Hafizullah Bin Ibrahim

adam

Ahmad Adam Bin Norhelme