KULLIYYAH OF ENGINEERING

DEPARTMENT OF MECHATRONICS ENGINEERING

Lab Report 10:

Systems Integration (Microcontroller, PLC and Computer Systems)

GROUP E

MCTA 3203

Semester 2, 2023/2024

MUHAMAD HAZIQ ISKANDAR BIN HASSAN NORDIN	2119327
MUHAMMAD ALIFF IQMAL BIN ZAINUN@ZAINUDDIN	2114805
MUHAMMAD NAZIM BIN AKHMAR	2114551
MUHAMMAD ARFAN BIN MOHD ZULKIFLI	2112945

Abstract

The goal of this project is to use Modbus transmission to link sensors, microcontrollers, and computers into a single system for industrial automation and control. There are three kinds of sensors in the system: an infrared (IR) sensor, a touch sensor, and a light-sensitive resistor (LDR). These are all linked to an Arduino, which acts as a Modbus slave. The Raspberry Pi is set up as a Modbus Master and is in charge of getting sensor data from the Arduino. Additionally, OpenPLC is used as an extra Modbus Slave to receive data from the Raspberry Pi and manage an output based on that data. This paper describes how to set up, configure, and show off the system, showing how it can use Modbus transmission to watch sensor data in real time.

Introduction

In the quickly evolving field of industrial automation, the capacity of a cohesive system to combine different components has become crucial for intelligent and productive operations. In this demonstration, the Modbus communication protocol is used to show how an OpenPLC, three

A Raspberry Pi, three outputs, and sensors can all be easily included into one system.
The Raspberry Pi serves as the Modbus Master in this experiment. It is responsible for interacting as a master device with an Arduino that is configured as a Modbus slave. The Arduino acts as a Modbus Slave and hosts three sensors whose data the Raspberry Pi wishes to read. Data from the sensors is sent to the Raspberry Pi using the Modbus protocol, which is essential for obtaining information from the actual world.

A Modbus Slave-configured OpenPLC system and the Raspberry Pi communicate at the same time. Three outputs are now under the control of the OpenPLC, which received the data from the Raspberry Pi. Additionally, the OpenPLC responds to sensor data received from the Arduino by controlling one of the outputs. The Raspberry Pi and the OpenPLC can communicate in both directions thanks to the Modbus protocol, which facilitates data interchange that is necessary for the dynamic control of the system.

This experiment illustrates the practical application of a distributed control system and emphasises the importance of utilizing well-established communication protocols such as Modbus to facilitate interoperability and simple integration among diverse industrial components. It is possible to create responsive, intelligent systems with real-time data processing, control, and acquisition capabilities using Raspberry Pi, Arduino, and OpenPLC.

Materials and Equipment

1) Raspberry Pi

2) Arduino Uno R3

3) Jumper Wires

4) LED

5) Resistor

6) Thermistor

7) LDR sensor

8) Tilt sensor

) Breadboard

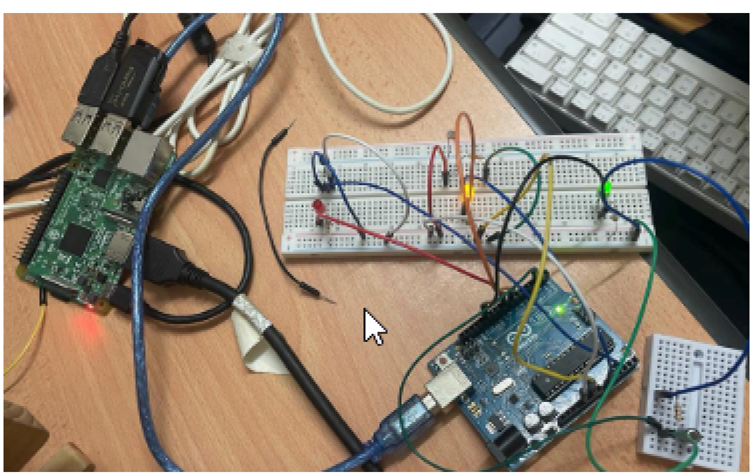
10) USB cable for Arduino Uno R3

11) HDMI cable

Experimental Setup

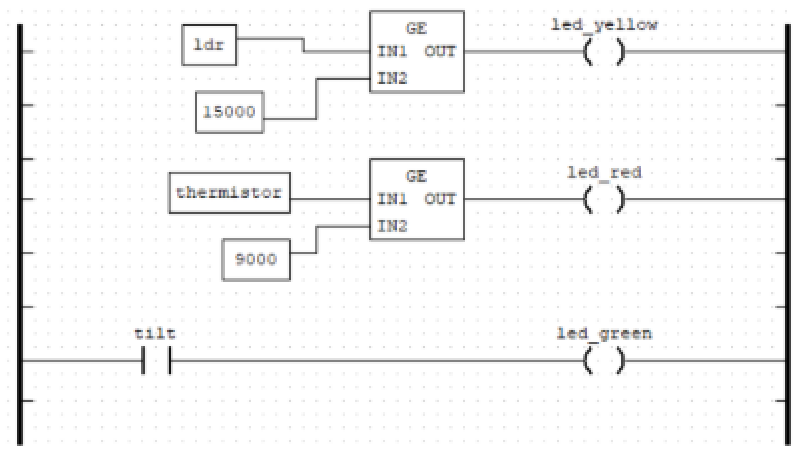
1. Raspberry Pi was connected to the Arduino as well as a monitor for display.

2. All the sensors and the output were connected to the Arduino as the diagram below:



3. A ladder diagram has been constructed as diagram below:

#	Name	Class	Type	Location	Initial Value	Option	Documentation
1	ldr	Local	INT	%IW0			
2	thermistor	Local	INT	%IW1			
3	tilt	Local	BOOL	%IX0.0			
4	led_yellow	Local	BOOL	%QX0.0			
5	led_red	Local	BOOL	%QX0.1			
6	led_green	Local	BOOL	%QX0.2			



Methodology

1. Sensors (name of the sensor that we used) are selected, and Modbus RTU serial protocol has been chosen for communication to establish a master-slave connection

2. Sensors are attached to the breadboard and connected to Arduino and Arduino-Raspberry Pi communication is established.

3. Arduino is programmed to read the sensors' data and Modbus communication is implemented for data transmission to Raspberry Pi.

4. The ladder diagram is developed with OpenPLC by using graphical symbols to depict logical control sequences for operating and managing the system components (sensors, Arduino, Raspberry Pi) based on input conditions and defined actions.

5. OpenPLC Runtime is installed in the system to demonstrate the system's ability to monitor all the sensors’ data in real-time and validate the data accuracy.

Results

We have effectively developed a system integrating a thermistor, LDR sensor, tilt sensor, Raspberry Pi, and Arduino, utilizing the Modbus communication protocol. Additionally, we have implemented real-time monitoring of all sensor data through the OpenPLC Runtime. As you can refer in the videos:

1. When a higher temperature is applied to the thermistor, the red LED illuminates indicate that it is true and the corresponding value in the OpenPLC Runtime increases. As the temperature returns to normal, the LED turns off, signifying a false status.
2. When LDR is exposed to the light, the yellow LED illuminates indicate true status and the corresponding value increases in the OpenPLC Runtime. As the LDR is covered or not exposed to the light, the LED turns off, signifying a false status and the value becomes negative.
3. When the tilt sensor is in the normal position (upright) the green LED illuminates and it appears true in the OpenPLC Runtime, then when the sensor is tilted, the LED turns off, signifying false status as shown in the OpenPLC Runtime.

Discussion

Our goal in this experiment is to use a microcontroller, a computer, and a variety of sensors to create a complete sensor system. Simple sensors including a thermistor, an LDR, and a tilt sensor were employed as inputs. We merely used basic multicoloured LEDs for the outputs, so each input will produce an output with a different colour. We thought we would only need to add one input with contact when we set up the ladder diagrams using OpenPLC Editor to deliver the command to the microcontroller, an Arduino UNO, but for some reason the sensors are unable to supply the value needed to activate the outputs. However, following several discoveries, we simply need to use variable in place of touch in order for the sensors to read and provide the data.

In order to turn on the yellow LED for the thermistor when the reading hit a predetermined level, we employed a comparison Greater Than block. However, the issue here is that we were unable to get

OpenPLC Runtime to display the actual temperature value; instead, it displays the thermistor's raw reading. We attempted to convert the figure into the actual temperature using an arithmetic block, but for some reason it didn't work. We utilised a comparison Greater Than block to turn on the Green LED for the LDR as well, and as the value can be adjusted for both brightness and darkness, there are no issues. Lastly, for the tilt sensor, we just directly use normal contact as an input to activate the Red LED and it has no problem as well since the type of value is just BOOL compared to other sensors which are INT type.

Conclusion

It is possible to describe this experiment as a successful integration of the thermistor, LDR, and tilt sensor as well as the Raspberry Pi as a Modbus Master and microcontroller as a Modbus Slave.
The system shows how Modbus enables easy communication between devices for control and monitoring, highlighting its usefulness in industrial applications. As students, ladder diagrams and communication protocols provide us with important insights into sensor systems. We can absolutely use this talent to further our careers in control and automation engineering.

Recommendations

Based on our experience experimenting in class and laboratory, we believe that code optimization is one of the most important things to focus on. Optimizing the Raspberry Pi and Arduino Uno R3 will directly make the entire system run smoothly. Adding comments to each code helps in terms of understanding the code much better and avoiding mistakes during runs of the experiment especially for students who just started learning programming. Next, the recommendations that will help improve these lab projects are fault tolerance and error handling. Improvising the system's robustness by incorporating fault-tolerant mechanisms and appropriate error handling. Also, provide simple and clear wrong parts and feedback to the user when the system is incorrect or the sensor malfunctions

References

1. O. (2022, October 22). Basics 03: Programming an Arduino Board with OpenPLC. YouTube. https://www.youtube.com/watch?v=d39Al4w1sL0

2. E. S. (2020, August 21). Tilt Sensor with Arduino. YouTube. https://www.youtube.com/watch?v=Uve-nEE1M-4

3. E. M. E. I. M. (2023, June 20). DIY Arduino PLC - How to Use the Analog Input in OpenPLC. YouTube. https://www.youtube.com/watch?v=c1HVic0NWMU

STUDENT'S DECLARATION

Certificate of Originality and Authenticity

This is to certify that we are responsible for the work submitted in this report, that the original work is our own except as specified in the references and acknowledgment, and that the original work contained herein has not been untaken or done by unspecified sources or Persons.

We hereby certify that this report has not been done by only one individual and all of us have contributed to the report. The length of contribution to the reports by Each individual is noted within this certificate.

We also hereby certify that we have read and understand the content of the total report and no further improvement on the reports is needed from any of the individual contributor to the Report.

We, therefore, agreed unanimously that this report shall be submitted for marking and this final printed report has been verified by us

Signature: haziq	Read
Name: Muhammad Haziq Iskandar bin Hassan Nordin	Understand
Matric Number: 2119327	Agree
Signature: aliff	Read
Name: Muhammad Aliff Iqmal bin Zainun @Zainuddin	Understand
Matric Number:2114805	Agree
Signature: nazim	Read
Name: Muhammad Nazim bin Akhmar	Understand
Matric Number: 2114551	Agree
Signature: arfan	Read
Name: Muhammad Arfan Bin Mohd Zulkifli	Understand
Matric Number: 2112945	Agree