Mechatronics System Integration (MCTA3203)

Week **10**: Systems Integration - Microcontroller, PLC and Computer Systems (ver 2.0)

Modbus Protocol

Overview

This module introduces practical system integration involving microcontrollers (Arduino), computers (Raspberry Pi), sensors, and actuators, using the Modbus RTU protocol over USB. In this setup, Arduino functions as a PLC simulator, interfacing with sensors and actuators, while Raspberry Pi acts as a Modbus Master, collecting data and issuing control commands.

What is the Modbus Protocol?

Modbus is a request-response protocol implemented using a master-slave relationship. In a master-slave relationship, communication always occurs in pairs—one device must initiate a request and then wait for a response—and the initiating device (the master) is responsible for initiating every interaction. Typically, the master is a human machine interface (HMI) or Supervisory Control and Data Acquisition (SCADA) system and the slave is a sensor, programmable logic controller (PLC), or programmable automation controller (PAC). The content of these requests and responses, and the network layers across which these messages are sent, are defined by the different layers of the protocol.

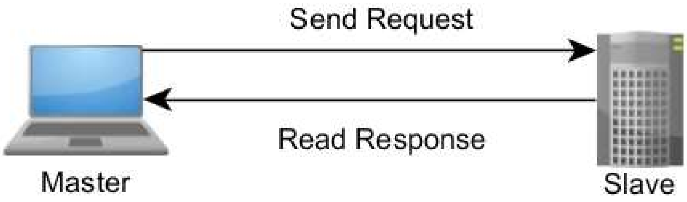

Fig. 1. A Master-Slave Networking Relationship

Objective

By the end of this module, students will be able to:

- Understand the architecture of a simple integrated automation system
- Implement Modbus RTU communication over USB between Raspberry Pi and Arduino
- Configure Arduino as a Modbus Slave
- Monitor and control sensors and actuators in real time
- Simulate PLC logic using Arduino with relay/actuator control

System Architecture

- Raspberry Pi (Modbus Master)
  - ◦ Collects sensor data from Arduino via USB
  - ◦ Runs Python script / Node-RED / SCADA
  - ◦ Sends control signal (if needed) to actuator
- Arduino Uno (Modbus Slave & PLC Simulator)
  - ◦ Reads data from 3 sensors (analog/digital)

  - ◦ Controls output (relay + water pump or actuator)
  - ◦ Communicates with Pi via USB serial (Modbus RTU)
- Sensors:
  - ◦ Limit switch – digital input
  - ◦ MPU6050 accelerometer/gyro – I2C, analog motion data
  - ◦ TF-Luna LiDAR sensor – distance measurement
- Actuator:
  - ◦ Relay module controlling micro water pump or linear actuator
- Communication:
  - ◦ Modbus RTU over USB

System Requirements & Configuration
Hardware:

- Raspberry Pi (any model with USB)
- Arduino Uno
- Sensors: Limit switch, MPU6050, TF-Luna
- Actuator: Relay + Water pump / linear actuator
- Power: External 5V for pump
- Cables: USB A to B for Arduino, jumper wires

Software:

- Raspberry Pi: Python + `minimalmodbus` / `pymodbus`, or `Node-RED`
- Arduino IDE + Modbus RTU slave library (e.g., `SimpleModbusSlave` or `ModbusSlave`)
- Optional: OpenPLC Runtime for control simulation

Project Goal:

Develop a fully integrated Modbus-based system with sensors and actuators using **Arduino as the PLC**, communicating with **Raspberry Pi as the master**.

Requirements:

Component	Description
3 Sensors	Connect to Arduino. Send readings to Raspberry Pi via Modbus.
Arduino Uno	Acts as Modbus Slave. Reads sensor data. Controls output.
Raspberry Pi	Modbus Master. Sends requests and receives sensor data.
Relay + Pump	Controlled by Arduino based on sensor input.
Modbus RTU	Communication over **USB serial**.
Monitoring	Real-time sensor data displayed on Pi (via Python script or Node-RED dashboard).

Example Communication Flow

1. Arduino reads sensors (limit switch, MPU6050, LiDAR)
2. Arduino responds to Modbus requests over USB
3. Raspberry Pi (Python/Node-RED) polls Arduino regularly

4. Raspberry Pi logs data / triggers actuator command
5. Arduino receives control signal, activates relay → pump

Deliverables

- Fully wired system with working USB serial communication
- Real-time monitoring dashboard (e.g., Node-RED on Raspberry Pi)
- Demonstration video or live test showing:
    - Sensor data updates
    - Pump/actuator activation via relay
- Source code:
    - Arduino sketch (Modbus Slave)
    - Raspberry Pi master script (Python or Node-RED flow)

Key Concepts Reinforced

- Modbus RTU Protocol (request/response)
- USB serial communication with microcontrollers
- PLC-like logic implementation on Arduino
- Interfacing sensors and actuators
- Data acquisition and visualization (SCADA/HMI-like)

References

[1] NI Modbus Protocol Guide
https://www.ni.com/en/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/the-modbus-protocol-in-depth.html

[2] OpenPLC Runtime:
https://autonomylogic.com/docs/installing-openplc-runtime-on-windows/

[3] 3. Modbus Python Tutorial:
https://pymodbus.readthedocs.io

[4] 4. Node-RED Modbus Setup:
https://flows.nodered.org/node/node-red-contrib-modbus

[5] 5. Arduino Modbus Library:
https://github.com/smarmengol/Modbus-Master-Slave-for-Arduino