Experiment 0
Communication

In this experiment, we will look at different communication techniques employed to transmit various signals on the IoT Board.

Background

All the signals on the IoT Board can be classified either as analog signals or digital signals. An analog signal can be a current or voltage or any type of physical-parameter-based signal that varies continuously with time. A digital signal on the other hand is a sampled and quantized sequence of such an analog signal. The amplitude of the signal is directly related to the physical parameter the signal is based on. Whereas for the digital signal, the amplitude varies between two discreet quantization levels. Some of the devices such as the potentiometer on the IoT board work with an analog signal while most sensor modules on the board transmit data digitally. The analog devices usually transfer a signal via a single line. The microcontroller has a built-in Analog to Digital Converter (ADC) which quantizes these analog signals to a digital signal. After processing the data, a user can send a signal which originates as a digital signal from the processing pipeline of the microcontroller. This signal will be converted back to an analog signal by using the Digital to Analog Converter (DAC) onboard the microcontroller so that it can be received by the analog device. Digital signals can also be transmitted via a single wire to a GPIO pin on the microcontroller like analog signals. Different communication protocols were introduced later to enhance the experience of communicating digitally like I2C, SPI, USB, PCIE, etc. These protocols usually have an accompanying clock signal with the data signal.

Digital signal transmission involving the use of a common clock pulse between the transmitter and the receiver is called Synchronous transmission. In asynchronous transmission, data is transmitted without a clock pulse. Digital data is transmitted in a group of bits. In synchronous transmission, the grouped bits are called frames or blocks. In asynchronous, they are in the form of bytes or characters. Each byte in asynchronous transmission has start and stop bits attached to the ends of the packet.

Since digital data is usually transmitted in groups of bits, these bits can either be transferred in serial or parallel. In serial communication, bits are transmitted one after the other over a single data line. Common examples of serial transmission protocols include USB, I2C, SPI, etc. Whereas in Parallel communication all the bits are transmitted at the same time on multiple data lines. Since all the bits are transmitted at the same time, applications that require a huge amount of data transmitted at high speeds utilize parallel communication protocols. PCIE, and ATA are some examples of such protocols. Most computer architectures utilize parallel communication to transfer data between the main processing complex and the memory like between CPU and RAM.

Digital data can be transmitted on multiple lines or a single line as we can observe in the previous paragraph. We can classify the transmission of either analog or digital signals by the number of transmission channels used in the protocols. Simplex mode involves sending both the transmitting and receiving signals on a single channel/line. Half-duplex and Full-duplex modes utilize two channels for transmitting data. One channel is for transmission and the other is for receiving. Half-duplex can only send a single signal at a time on either channel. So the transmission channel remains unused if the receiving channel is being used. In Full-duplex, both the transmitting and the receiving channels can be utilized at the same time. A radio station transmitting a broadcast is an example of Simplex mode. Walkie-Talkies, I2C, or WiFi for example are Half-duplex modes of transmission. A telephone call for example can be considered a Full-duplex mode of communication.

Experiment Set-up: Configuration

Part 1

In part 1 of this experiment, we look at communication between the Node MCU and one of the sensors on the IoT Board. We will connect two sensors, namely the Ultrasonic sensor and the Environment sensor. Unmount the Node MCU from its position and mount it on the breadboard. Then unmount the Ultrasonic sensor and place it on the breadboard. Now connect the pins on the ultrasonic module to the GPIO pins of the Node MCU. The Ultrasonic sensor utilizes two trigger pulses as input and output signals. After observing the data transmission between the Node MCU and the Ultrasonic sensor, disconnect them. Unmount the Ultrasonic sensor and mount it back to its corresponding position on the IoT Board. Similarly, connect the Environment sensor by unmounting it from its position and then placing it on the breadboard. Now observe the communication between the sensor and the microcontroller.

Part 2

In part 2 of this experiment, we look at communication between two Node MCUs. Team up with another group to pair your Node MCUs together. Unmount the Node MCU and place it on the breadboard. Using a pair of wires connect the two Node MCUs on different IoT Boards. You can connect the wires to the I2C pins near the breadboard. Upload the program given in the course material. Press a button on each IoT Board to send a signal between each microcontroller.

Instructions

Part 1

1. Unmount both the Node MCU and Ultrasonic Sensor and place them on the breadboard.
2. Look at the board schematic and determine the pin connections for both of them. Connect the Trig and the Echo pins of the Ultrasonic sensor to the GPIO pins of the microcontroller.
3. Open Arduino and click on “File” from the Menu bar. Click on “Open” to open the “ultra.ino” file. The “ultra.ino” file can be found in the course material for this module. The IDE now loads the program. Edit the code to add the pin connections.
4. Once you’ve made changes to the code, flash the program. Refer to the IoT Board Manual for flashing instructions.
5. Open the Serial Monitor from the dropdown menu under “Tools” on the Menu bar. Refer to the IoT Board Manual for setting up the Serial Monitor window. In the serial monitor, the distance value measured is refreshed constantly.
6. After observing the communication, disconnect the power to the Node MCU and disconnect the Ultrasonic sensor.
7. Now unmount the Environment sensor from its position and place it on the breadboard. Connect the I2C pins and the voltage pins accordingly.
8. Install the BME280 library and the Adafruit_sensor library from the GitHub links provided in the References section. Download the zip files and extract the folders. Rename the folder to remove the “-master” bit and move the folders to the default library folder.
9. Open the Arduino IDE and click on “File” from the Menu bar. Click on “Open” to open the “env.ino” file. The “env.ino” file can be found in the course material for this module. Now the IDE displays the code that needs to be flashed onto the microcontroller.
10. Now compile the code and look for any errors. Then proceed to upload the code onto the microcontroller. Refer to the IoT Board Manual for flashing instructions.
11. Open the Serial Monitor from the dropdown menu under “Tools” on the Menu bar. Refer to the IoT Board Manual for setting up the Serial Monitor window.

12. In the Serial Monitor Window, the Node MCU will now display temperature and humidity values that the sensor measures.
13. After observing the communication, disconnect the power to the Node MCU and disconnect the Environmental sensor.

Part 2

1. Pair up with a group and decide which Board will be named MCU1 and MCU2. Unmount the Node MCUs from their positions and place them on the breadboard.
2. Unmount all the sensor modules from the IoT Board designated as MCU1. Make sure to replace them in their respective positions after the experiment.
3. Connect the Node MCU to your PC. Open the Arduino IDE and click on “File” from the Menu bar. Click on “Open” to open the “MCU1.ino” file. The “MCU1.ino” file can be found in the course material for this module. Now the IDE displays the code that needs to be flashed onto the microcontroller.
4. Compile the code and upload the code onto the microcontroller. Refer to the IoT Board Manual for flashing instructions.
5. Similarly, flash the second Node MCU with the “MCU2” code.
6. Now place both the Node MCUs on their respective breadboards.
7. Using jumper wires connect the I2C pins of the Node MCU to the I2C pins on the IoT board. Perform this connection only on the board designated as MCU1.
8. Now use the other I2C pins on MCU1 to connect the MCU2 directly on the breadboard. Do not connect the jumper to the I2C pins near the breadboard on the MCU2 designated IoT Board. Connect the I2C wires from the MCU1 board to the Node MCU placed on the breadboard of the MCU2 board.
9. Make sure the connections are correct and then power both the microcontrollers ON.
10. Now pressing the buttons on the MCU1 marked board will send a signal to the MCU2 marked board. This signal will be visible in the serial monitor of the MCU2 marked board. Similarly, pressing the buttons on the MCU2 should send a signal to the MCU1 marked board.
11. Observe the messages on the Serial Monitors of both IoT Boards.

☞ Deliverables

Demonstration:

1. Record a video demonstration explaining the outcome of the experiment. Refer to the title page for a brief description of the expected outcome. Make sure you talk over all observations and the video is presentable. Also, don't forget to show the data updating in real time on the serial monitor and the OLED screen.
   Can you connect the Ultrasonic sensor to any GPIO pin on the Node MCU? Explain your reason. Also similarly which pin would you use to connect the Potentiometer and the Heart-rate sensor? What is the message you are receiving in MCU2 from MCU1?

References and Further Reading

[1] https://github.com/adafruit/Adafruit_BME280_Library/archive/master.zip
[2] https://github.com/adafruit/Adafruit_Sensor