Electrical Engineering and Computer Science
Colorado School of Mines

Prepared by Vibhuti Dave and Tyrone Vincent

EENG 498: SEED Lab
Spring 2016
Exercise 3: Sensors and Processor Communication

- **Introduction**
  The purpose of this exercise is to interface sensors with the Arduino, and establish communication between two arduinos. At the end of this exercise, you should be able to read data from the sensors, interpret the data and provide an output based on the data. You should also be able to send data from one Arduino to another.

- **Ultrasonic Distance Sensor**
  Ultrasonic sensors can be used to measure distance. It is also a good alternative to detect objects/obstacles. These sensors emit high-frequency sound pulses which reflect back when they strike an object. The sensor can compute the distance to the target based on the time-span between the emitting signal and the signal that echoed back. We will interface the Ultrasonic Ranging Module HC – SR04 with the microcontroller. The sensor will be placed on the robot to detect obstacles. There is an Arduino Library called NewPing to help interface with this sensor.

- **RGB and Gesture Sensor**
  The robot will need to detect a finish line to know that it has arrived at its destination. We will use SparkFun RGB and Gesture Sensor - APDS-9960 to detect a colored line on the floor. It can also be used to control the motion of the robot using hand gestures. The APDS-9960 has built-in UV and IR blocking filters as well as diodes sensitive to different directions. Sparkfun also provides the library and example sketches to go along with it.

- **Gyro**
  A gyrometer (gyro) is a device which can measure angular velocity, or the rate at which an object is rotating. The LPY503AL uses MEMS technology (Microelectromechanical systems) to implement a gyro. The output is an analog voltage. No special library installation required.

- **Interrupts**
  The events described above are asynchronous events; meaning they can occur at any time while your robot is balancing and moving forward. Interrupts are an efficient way of detecting such asynchronous events so your code doesn’t have to waste time keeping an eye out for these events and can keep doing what its doing. When the microcontroller gets interrupted, the code will stop what its doing and execute an Interrupt Service Routine (ISR) instead. Once the ISR completes execution, it can go back to doing what its doing.

  1. How many external interrupts are available on the Uno? Which pins are they available on?

2. What built-in Arduino function can you use to configure external interrupts on the microcontroller? This is very convenient since it doesn't require you to configure the internal registers of the microcontroller to handle interrupts. Provide an example sketch.
3. The function from question 2 has its own overhead. Which registers in the microcontroller would need to be configured to handle interrupts if you wanted to avoid the overheard associated with the built-in Arduino function? Explain with example sketches.

- **Serial Communication**

  This is the mode of communication that the Arduino board uses to communicate with a computer or other devices. With serial communication the Arduino sends bits one by one, i.e., serially. All Arduino boards have a UART/USART port. In addition, digital pins 0 (RX) and 1(TX) also provide a means for serial communication. If pins 0 and 1 are used as RX and TX lines, then they cannot be used as digital inputs or outputs. The different commands available for serial communication can be found here. You will use the RX and TX lines on the Arduino board to connect two Arduinos together. There is also need for a common ground.

- **I^{2}C Communication**

  This protocol uses the master/slave configuration and also relies on serial communication. It is one of the most useful and widely used protocols because it allows multiple devices to communicate with one another using the same two pins. The devices are addressable with a unique address between 8 and 119. Analog pins 4 (SDA or serial data) and 5 (SCL or serial clock) are required to set up communication between two Arduinos.

  Master: The device that generates the clocks and initiates communication with the slaves.
  Slave: The device that receives the clock and responds to the master. Each slave will have a unique 7-bit address. It is recommended to have the master transmit and think of the slave as the receiver although roles can be reversed. If the slave is asked to transmit information, it can slow things down. If speed is not a concern, roles can be reversed and you can even establish two-way communication.

  Step 1: Master starts transmission by sending a start but followed by a 7-bit address of the slave it wishes to communicate with. The final bit specifies if it wants to read from or write to the slave. This is a total of 9 bits.
  Step 2: Slave responds with an active low ACK (acknowledge) signal.
  Step 3: Mater continues to transmit and slave continues to receive.
  The address and data byte are sent most significant bit first.
  A brief explanation of how this protocol works with Arduinos along with an example code for the transmitter and receiver can be found here.

  1. What is the maximum rate of data transfer that you can achieve with an I^{2}C setup?
  2. What Arduino library do you need to use to implement this?
  3. How many bytes can you transmit in one transaction?

- **Wireless Communication**

  There are several different options in this category. You can choose to go with Bluetooth, XBees, IR, etc. For the purpose of this exercise, we are going with a cost-effective solution of using nRF24L01

(RF transceiver) operating at 2.4 GHz. It is widely known for its ultra-low power consumption and high-speed data rate options.

1. What is the maximum rate of data transfer that you can achieve with the RF transceiver?
2. What Arduino library do you need to use to implement this?
3. How many bytes can you transmit in one transaction?

- **Tutorials (Click on the links for basic tutorials)**

1. Complete Guide to HC-SR04
2. RGB and Gesture Sensor

- **Challenges**

1. The micro controller on the Arduino Uno board has 3 timers built into it. Use this feature along with interrupts and the ultrasonic sensor to build a parking sensor. Print the distance of the car from the curb on the serial monitor. Increase the rate at which an LED flashes, the closer the car gets to the curb. Use of timers and interrupts is mandatory to keep the code and design efficient.
2. Interface the APDS 9960, to detect color and proximity. Have the microcontroller display messages on the serial monitor when it detects the color red. If it detects the color red between 5 and 10 cm, it displays the message to slow down. Once it detects the color red closer than 5cm, it should display the message to stop. For distances greater than 10 cm, the program should display a message to keep going.
3. Mount the gyro to a frame with an Arduino, and wire up the gyro. Note that the gyro runs on 3.3V. Write an Arduino program that will read the gyro analog output at a set rate. This program should have an initialization which will set up the Arduino correctly, including performing a high pass filter reset. Use serial communication to send the data to a computer, and save the data, including timestamp. Copy and paste the data in an excel spreadsheet and plot the data.
4. Use the I^{2}C protocol to transmit one byte of data from the master to the receiver. Scope out the SCL and SDA lines so you can see the transmission of one 8-bit data from the master to the slave. Print out the waveform and clearly mark the start bit, the 7-bit address, the acknowledge bit, the data byte and the stop bit.
   a. How many ms does it take to transmit one byte starting from the start to stop?
   b. How many data bytes can you send per second?
   c. Is there a way to increase the rate of data transfer. Show an example sketch of how this may be done.
5. When working with motors in Exercise 2, you were asked to calculate the velocity of the motors interfaced with the Arduino. Set up two Arduinos such that you can send motor velocity from one Arduino to another. One of Arduinos is the master and will transmit data, the other Arduino takes on the role of the slave to receive data. Received data should be printed to the serial monitor, and saved to an excel file.
6. Use two RF transceivers to repeat challenge 5. This will be useful for debugging purposes.

- **Deliverables**
  - Upload answers to questions in the handout
  - Upload well commented sketches for all challenge exercises.
- **Demonstrations**
  - Be prepared to demonstrate and answer questions for challenges 2, 3, 6.