Laboratory #6
Ultrasonic Sensor

Objectives:

- Implement a device driver to interface an ultrasonic sensor
- Measure the execution time of your function that handles the ultrasonic sensor
- Use OLED to display sensor measurements
- Interface a Bluetooth module to provide remote access for the sensor measurements
- Implement PIC32 sleep mode for power saving (5% extra credit)

Files provided on Blackboard:

- Ultrasonic sensor user manual
- OLED user manual
- UART device driver (`UART.h` and `UART.c`)
- OLED device driver (`SH1106.h` and `SH1106.c`)
- `main.c` (incomplete program file)

Hardware Configuration:

The I/O shield used in the previous labs comes with several on-board devices some of which you have already used before. In this lab we will be using some of the header pins provided on the board to interface external devices, such as ultrasonic sensor, OLED, and Bluetooth module. See Figure 1 below for the correct mounting of these devices on the I/O shield.

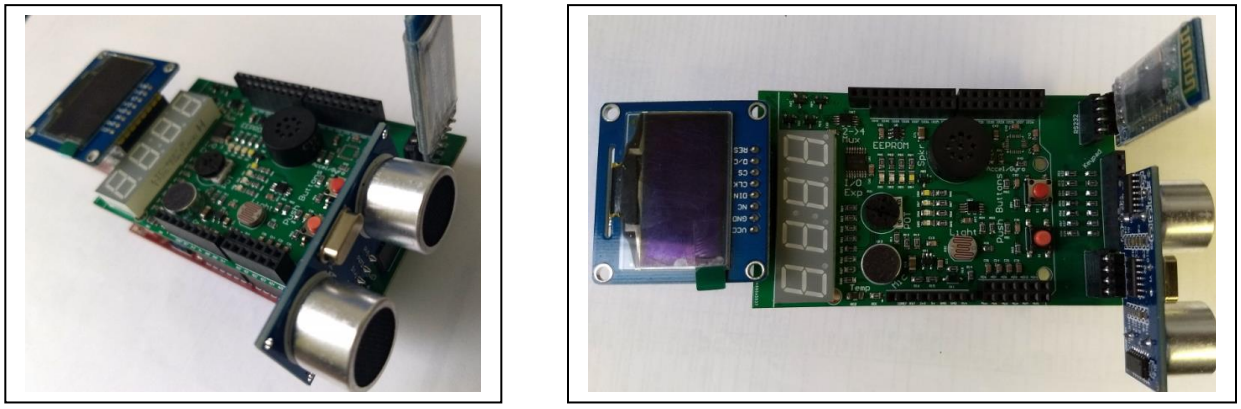

Figure 1. Hardware connection of ultrasonic sensor, OLED, and Bluetooth module

NOTES:

a) To avoid damage to the I/O shield and the external devices, it is critical that you be careful in properly installing the devices on the board (check the Vcc, GND, and other pins).

b) Since the ultrasonic sensor draws more current than can be supplied by the chipKIT voltage regulator driven by the programmer/debugger USB interface, you need to connect additional power via the second USB port of the chipKIT board or from a 9V power supply via the barrel-jack power connector of the board.

Tasks

1) Ultrasonic Sensor Interfacing

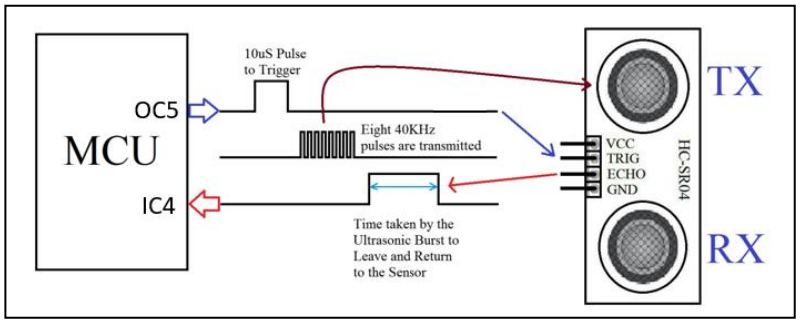

Figure 2. Ultrasonic sensor operation

Ultrasonic sensor is widely used in robotics and other applications for measuring distance to an object in front of the sensor. Its principle of operation is based on the propagation of sound waves and the fact that such waves reflect back (echo) when the waves encounter solid objects. With the knowledge of speed of sound in air and by measuring the time it takes for the sound waves to complete the round trip travel from the sensor to the object and return back to the receiver, one can calculate the distance to the object. Figure 2 shows how this operation works. The ultrasonic sensor device you will be using in this lab assignment is the HC-SR04, which is commonly available and used for many applications, including obstacle detection and avoidance in robot navigation. For your reference, the user manual for this device is provided on Blackboard.

To interface the ultrasonic sensor to your PIC32 and implement a device driver to read the sensor measurement, you will need to utilize the timer Output Compare (OC) and Input Capture (IC) modules. The trigger pin (TRIG) of the sensor is connected to OC5 pin and the ECHO pin of the sensor is connected to the IC4 pin. It is important to implement precise timing to generate the 10 microsecond pulse width trigger signal for the sensor and to accurately measure the pulse width on the echo pin for accurate distance measurement. Refer to Lect6b-PIC32-Timers-IC-OC, the PIC32 MCU datasheet, and the PIC32 peripheral library documentation available on Blackboard for information on how to configure and use the input capture (IC) and output compare (OC) functions of the PIC32 MCU.

Note that as you will be configuring your input capture module to measure the pulse width (in units of time) of the ECHO pulse coming from the sensor, you will need to convert this time measurement to actual distance from the sensor to the target. For this conversion you need to use information about the speed of sound in air, which you can assume to be 340 m/s.

You will need to implement a function with the following prototype for measuring distance to target objects using the ultrasonic sensor:

```
float  sonarDistance(void);
```

To test your program you will also need to implement a main program that calls the `sonarDistance()` function and displays the result on the OLED.

2) OLED

In this lab, for displaying the sensor measurements you will be using a small 1.3", 128x64 pixels resolution OLED. The user manual for this display is available on Blackboard. The display uses the Serial Peripheral Interface (SPI) for its communication with the PIC32 MCU. Library utility is provided on Blackboard that you can include in your project to facilitate the use of the OLED for your application. A simple `main()` function demonstrates how you can initialize the OLED and access its functions to print information on the display.

3) Measuring Accuracy of your Ultrasonic Sensor

Once you are able to receive your ultrasonic sensor measurements on the OLED, in this task you will collect data for target objects placed at varying distances and compare your measurement against a professional laser based distance measuring instrument. Create a table that shows the measurement values from your ultrasonic sensor and the laser sensor, with distances ranging from 20 cm to 200 cm, at intervals of 10 cm. Also, create an additional column in your table that shows the distance measurement error (assuming the laser instrument to give a relatively accurate reference). Explain your observation about the accuracy of your ultrasonic sensor, and whether it gets affected with distance.

4) Measuring Execution Time

For this task, you are required to setup an experiment for measuring the execution time of your device driver function (from task 1 above) for interfacing to the ultrasonic sensor.

Hint: Use the PIC32's timer services you are now familiar with for your time reference.

After you setup the method for measuring the execution time of your *sonarDistance()* function, create an experiment to collect data for execution time of the function when measuring different distances, ranging from 10 cm to 200 cm, with increments of 10 cm. Present your result in a table. Next, analyze the data and discuss how the execution time of the *sonarDistance()* function is affected by the distance you are measuring.

5) Bluetooth Communication

4.1) Basic Operation

To provide wireless communication access to your application running on the PIC32, you will be using HC-06 Bluetooth module connected to your MCU. This is a widely available, low-cost, and easy way to create a wireless interface to a microcontroller. An online document with Arduino example on how to use the Bluetooth module is available at the following link:
http://wiki.sunfounder.cc/index.php?title=Bluetooth_Transceiver_Module_HC-06

The Bluetooth module interfaces to the MCU using a UART port. The PIC32MX320F128H module on the chipKIT UNO32 kit comes with two UART modules. UART1 is already wired to the USB port via a UART-to-USB adapter. This is typically used for serial communication with a PC. For the Bluetooth interfacing you will be using the UART2. The receiver (U2RX) and transmitter (U2TX) pins of UART2 are available on the chipKIT UNO32 pins 39 and 40, respectively, and are brought out to the female headers for connecting the Bluetooth module.

You are given on Blackboard the UART driver programs for interfacing the PIC32 MCU to a PC serial terminal and also a wireless Bluetooth module. The functions provided are used for initializing the UART modules, and transmitting and receiving data through the serial ports.

4.2) Bluetooth enabled Android device or laptop

You will be implementing a program for your PIC32 that will interface to a Bluetooth enabled device (such as an Android Smartphone or a laptop with built in Bluetooth feature) to provide a remote access to your ultrasonic sensor.

If you have access to an Android device, it is recommended to download and use a Bluetooth Terminal application, such as the "Serial Bluetooth Terminal" or similar app available on Google Play:

https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal&hl=en.

You can send and receive messages between your device and the MCU using such an app. The UART driver functions provided on Blackboard assume that your app communicates at 9600 baud rate; which is the default setting for the "Serial Bluetooth Terminal" app.

You will provide three operating modes for the ultrasonic sensor as follows. The mode selection will be done by sending one letter commands from your Bluetooth enabled device, as follows:

Character command	Action on ultrasonic sensor
R	Run sensor (continuously refresh the sensor measurement)
P	Pause the measurement (stops updating the measurement)
S	CPU enters low-power sleep mode (5% extra credit)

Your program should accept both upper and lower case commands.

When the application is operating in run (R) or pause (P) modes, the measurements need to be displayed on both the OLED as well as the mobile device (over the Bluetooth interface).

If you choose to do the sleep mode (S) for the extra credit, you can use the UART receiver interrupt as a means to wake up the CPU from sleep. You can then send any character from your Bluetooth device to trigger the CPU to wake up and return into one of its normal operation modes (either of the run or pause modes).

Check-off and report:

Demonstrate your working program to the lab instructor and submit, via Blackboard, a lab report that follows the format and structure guidelines given in "Laboratory and Project Report Guidelines" available on Blackboard. Your report needs to include: Title page, Table of Contents, Objectives, Hardware, Program Source Code, Experiment (for the execution time measurement) with Data and Analysis, and finally Conclusions. For task 3, present your data table for the distance measurements and give analysis on the accuracy of the ultrasonic sensor. For task 4, include a section that explains the method you used for measuring the execution time of the `sonarDistance()` function, give a table with data you collected, analyze the data and give your observation.