# *Lab 3 (part 2): Reading of Sensors*

## Preparation

***Reading***

Lab Manual:

*Chapter 4 - The Silicon Labs C8051F020 and the EVB* (sections on the *C8051 timer functions, interrupts, and the programmable counter array*)

*Chapter 5 - Circuitry Basics and Components (The buffer)*

*Chapter 6 - Motor Control (DC Motors)*

## Objectives

***General***

Like Laboratory 3-1, this laboratory has individual assignments. Implement the sensor hardware and write code that can read the sensors. The ranger developers and the compass developers will write the code to read respective sensors.

***Hardware***

1. The respective developers will install ultrasonic ranger and electronic compass for data collection. Note, two members of the team will be using the ultrasonic ranger.

***Software***

1. Ultrasonic Ranger Partner 1– Reading the range - (1) request that the ranger starts an ultrasonic ping, (2) wait 65ms while the ranger "listens" for an echo [in some cases, waits of 80 or 100ms give more consistent results], (3) request that the ranger value be sent on the $I^2C$ bus.
2. Electronic Compass Partner 2 – Reading the direction - (1) obtain a reading and (2) control how often to get a reading.
3. Ultrasonic Ranger Partner 3– Reading the light sensor - (1) request that the ranger starts an ultrasonic ping, (2) wait 65ms while the ranger "listens" for an echo [in some cases, waits of 80 or 100ms give more consistent results], (3) request that the ranger value be sent on the $I^2C$ bus.

## Motivation

In the previous lab, pulse width modulation was introduced to change the speed of drive-motor, steering of servomotor, and blinking of an LED. In this lab, a program reads the electronic compass or the ultrasonic ranger, which will be used to control the steering servo, speed controller, and LED brightness in Lab 4.

## Lab Description & Activities

*Ultrasonic Ranger – Reading the distance (Partner 1)*

Read the specification on ultrasonic ranger and wire the ranger according to the given circuit diagram. Note that the ranger is to be mounted on the small board provided on the *Smart Car*. Don't move the rangers since those are also used in 3 other sections. The following pseudo-code describes the main function to read from the ranger

```
Initialize everything
       start while (1) loop
          if 80ms has passed
             call ranger distance function
             print range - (printing every 80ms is more than needed)
```

1. Write a ranger function that reads the ultrasonic ranger.
2. Check the ranger for accuracy.
   - Hold an object above the car, measure the height with the ranger and with a tape measure.
   - Repeat for several distances
   - Repeat with a piece of carpet
   - What is the closest an object can be and you still get reliable readings?
3. Check the ranger for sensitivity to objects off to the side. Crudely estimate the angular sensitivity of the ranger. Record the information of ranger sensitivity in your lab notebook and have a TA verify the performance of the ranger.

*Electronic Compass – Reading the direction (Partner 2)*

Read the specification on electronic compass and wire the compass according to the given circuit diagram. Note that the compass is to be mounted on the small board provided on the *Smart Car*. Don't move the compasses since they are also used in 3 other sections. Note the orientation of the module. It must be mounted on the small protoboard toward the front of the car and hanging out in front as much as possible to avoid stray fields from other iron parts. Its reference will then be in line with the car. So when the car is aligned North, the sensor should read about 0°. The room has a fair amount of magnetic interference. The direction of 'North' will depend on where you are located, especially if you are near an active power conduit.

The following pseudo-code describes the main function to read from the ranger

```
Initialize everything
       start while (1) loop
          if 40ms has passed
             call compass function
             print direction - (printing every 40ms is more than needed)
```

1. Write a compass function that reads the electronic compass.
2. Check the compass for accuracy.
   – Use the 'mechanical' compasses to find north at your station and point the car in that direction.
   – Rotate the car, reading the compass at various angles
   – Verify that the compass returns values from 0 to 360º (0 to 3600).
3. Record the results in you lab notebook and have a TA verify the results, even if you don't get any better accuracy than what you started with.

*Ultrasonic Ranger – Reading the light sensor (Partner 3)*

Read the specification on the ultrasonic ranger and wire the ranger according to the given circuit diagram. Note that the ranger is to be mounted on the small board provided on the *Smart Car*. Don't move the rangers since those are also used in 3 other sections. The following pseudo-code describes the main function to read from the ranger

```
Initialize everything
       start while (1) loop
          if 80ms has passed
             call ranger light sensor function
             print light value - (printing every 80ms is more than needed)
```

1. Write a ranger function that reads the ultrasonic ranger.
2. Check the ranger for accuracy.
   – Cover the ultrasonic ranger with your hand
   – Repeat, opening your hand to allow more light onto the sensor
   – What are the darkest and brightest light sensor readings?

*Electronic Compass – Reading the direction*

Read the specification on electronic compass and wire the compass according to the given circuit diagram. Note that the compass is to be mounted on the small board provided on the *Smart Car*. Don't move the compasses since they are also used in 3 other sections. Note the orientation of the module. It must be mounted on the small protoboard toward the front of the car and hanging out in front as much as possible to avoid stray fields from other iron parts. Its reference will then be in line with the car. So when the car is aligned North, the sensor should read about 0°. The room has a fair amount of magnetic interference. The direction of 'North' will depend on where you are located, especially if you are near an active power conduit.

The following pseudo-code describes the main function to read from the ranger

```
Initialize everything
       start while (1) loop
          if 40ms has passed
             call compass function
             print direction - (printing every 40ms is more than needed)
```

1. Write a compass function that reads the electronic compass.
2. Check the compass for accuracy.
   - Use the 'mechanical' compasses to find north at your station and point the car in that direction.
   - Rotate the car, reading the compass at various angles
   - Verify that the compass returns values from 0 to 360º (0 to 3600).
4. Record the results in you lab notebook and have a TA verify the results, even if you don't get any better accuracy than what you started with.

## Software - Ultrasonic Ranger – Reading Distance Task (Partner 1)

The pseudo-code of the ranger function, which reads the ultrasonic ranger, is:

```
while (1)
{   if 80ms (flag) have passed
    {
       read the ranger
       start a ping
       reset the 80ms flag
       print the range
    }
}
```

**Notes:**

1. The time lag of 80 ms is used because we have already created a counter with overflows every 20 ms using the PCA timer overflow interrupts; 4 PCA timer overflows yield 80 ms.
2. The first reading will be wrong because we haven't requested a ping, but after 80ms we will have valid readings. The gondola doesn't travel very far in 80ms, the car travels even less.
5. To read data:

```
unsigned int ReadRanger()
{
   unsigned char Data[2];
   unsigned int range =0;
   unsigned char addr=0xE0;           // the address of the ranger is 0xE0
   i2c_read_data(addr, __, Data, _);  // read two bytes, starting at reg 2
   range = (((unsigned int)Data[0] << 8) | Data[1]);
   return range;
}
```

4. To start a ping with the result in cm:

```
Data[0] = 0x51;                      // write 0x51 to reg 0 of the ranger:
i2c_write_data(addr, ___, Data, __); // write one byte of data to reg 0 at addr
```

5. To create a flag that indicates that 80ms have passed, put the following into the PCA ISR:

```
void PCA_ISR ( void ) __interrupt 9
{
   if (CF)
   {
       ...
       r_count++;
       if(r_count>=4)
       {
           new_range=1;  // 4 overflows is about 80 ms
           r_count = 0;
       }
   }
}
```

6. The ‘new_range’ is the flag used to determine if a new range is ready to be read:

```
if (new_range) { clear new_range, read range, start a new ping}
```

## Software - Electric Compass Task (Partner 2)

To read the compass you simply need to read registers 2 and 3 of the compass module. Both this sensor and the ranger are set so that if you read register 2 and then request another read, you will get register 3. So, The pseudo-code of the compass function, which reads the electric compass, is:

```
unsigned int ReadCompass()
{
   unsigned char addr = _____; // the address of the sensor, 0xC0 for the compass
   unsigned char Data[2];      // Data is an array with a length of 2
   unsigned int heading;       // the heading returned in degrees between 0 and 3599
   i2c_read_data(addr, ____, Data, _);  // read two byte, starting at reg 2
   heading =(((unsigned int)Data[0] << 8) | Data[1]);  //combine the two values
                               //heading has units of 1/10 of a degree
   return heading;             // the heading returned in degrees between 0 and 3599
}
```

**Notes:**

1. The electronic compass is easy to use because it is always updating itself. Any 'read' commands will result in a reply of the last valid reading. It is however a tricky thing to use inside the LITEC lab because there are stray magnetic fields. The central column in the studio appears to be magnetized, and looks like the North Pole when you get close. There is a lot of steel reinforcing in the columns and in the floor. So while it is easy to take magnetic readings, it isn't easy to really know which direction you are headed. This problem doesn't exist in the Armory. The object then is to make the compass work as best as you can in this room, with the expectation that things will get easier later.
2. You need to consider how often and when to read the compass. The compass takes about 35 ms to complete a reading. Our controller will need to look at the time rate of change of the heading, (the turning rate of the gondola.) We will determine the time rate of change by taking the difference of two compass readings and effectively dividing by the time difference:

   ```
   rate of change = (heading change)/(time change).
   ```

   For the above reason, we don't want to take two heading readings within 35ms. If both readings occur within one actual sensor update, both readings will be the same and our code will assume the gondola isn't turning, while it may well be turning very rapidly. The solution suggested here is to wait 40ms between requests to the compass module for the present heading. 40ms is chosen because it is equal to time of 2 PCA timer overflows:

   ```
   void PCA_ISR ( void ) interrupt 9
   {
      if (CF) {
             ...
             h_count++;
             if(h_count>=2)
             {
                new_heading=1;  // 2 overflows is about 40 ms
                h_count = 0;
             }
       ...
      }
   }
   ```

In the main program, you will get a compass reading only if new_heading is set high in the PCA interrupt function above. The 'new_heading' is a flag that is set in the PCA ISR and cleared when you do a compass reading.

```
if (new_heading)  // enough overflows for a new heading
{
   heading = read_compass();
   printf(...)   // print heading
   /* Printing every compass reading will slow down the code significantly.
      Consider printing every 5th reading. */
   new_heading = 0;
}
```

## Software - Ultrasonic Ranger – Reading Light Task (Partner 3)

The pseudo-code of the ranger function, which reads the ultrasonic ranger, is:

```
while (1)
{   if 80ms (flag) have passed
    {
       read the ranger
       start a ping
       reset the 80ms flag
       print the light
    }
}
```

**Notes:**

1. The time lag of 80 ms is used because we have already created a counter with overflows every 20 ms using the PCA timer overflow interrupts; 4 PCA timer overflows yield 80 ms.
2. The first reading will be wrong because we haven't requested a ping, but after 80ms we will have valid readings.

6. To read data:

```
unsigned int ReadRanger()
{
   unsigned char Data[1];
   unsigned int light = 0;
   unsigned char addr=0xE0;           // the address of the ranger is 0xE0
   i2c_read_data(addr, __, Data, _);  // read one byte, starting at reg 1
   light = Data[0]
   return range;
}
```

4. To start a ping:

```
Data[0] = 0x51;                       // write 0x51 to reg 0 of the ranger:
i2c_write_data(addr, ___, Data, __); // write one byte of data to reg 0 at addr
```

5. To create a flag that indicates that 80ms have passed, put the following into the PCA ISR:

```
void PCA_ISR ( void ) __interrupt 9
{
   if (CF)
   {
      ...
      l_count++;
      if(l_count>4)
      {
         new_light = 1;  // at least 4 overflows is about 80 ms
         l_count = 0;
      }
   }
}
```

6. The ‘new_light’ variable is the flag used to determine if a new light sensor value is ready to be read:

```
if (new_light) { clear new_light, read range, start a new ping}
```

## Lab Check-Off: Demonstration and Verification

1. Complete the required entries in your lab notebook and present it to your TA.
2. Your TA may ask you to explain how sections of the C code or circuitry you developed for this exercise work. To do this, you will need to understand the entire system.
7. Ultrasonic Ranger Developer - How would you change the commands to receive the results in inches or msec? How would you read the range results out of the ranger?
4. Electronic Compass Developer - How would you change the commands to receive the results as a number between 0 and 255?

## Grading - Preparation and Checkoff

Prior to the starting the laboratory you must complete

1) The appropriate Worksheets (Worksheets 9).
2) The Pin-out form (Port, Interrupt, XBR0, PCA SMB initializations
3) The Pseudocode (Revision when finished)

When you are ready to be checked off, the TAs will be looking at the following items

4) That your project performs all the indicated requirements (defined above in ***Lab Description & Activities***)
5) Appropriately formatted and commented source code
6) Clean and neat hardware, with appropriate use of colors for source and ground connections

Additionally, you will be asked a number of questions. The questions will cover topics such as

7) Understanding algorithms in the code, identifying locations in the software that perform specific actions, understanding the hardware components, understanding the test equipment

The final item that will be included in the Laboratory grade is your

8) Notebook.

The above 8 items each have an individual contribution to your Laboratory grade. When completing this exercise make sure the compass and ranger remain mounted on the car to be available for use in the next class or open shop. **The compass and ranger should never be mounted on your own protoboard.**

## Writing Assignment - Lab Notebook

Your Lab Notebook must be kept up to date by recording the work you and your partner do in the lab. This should include pseudo-code for your system, data graphs, and a copy of your final C code for this lab. Further information on keeping a good Lab Notebook can be found in *Writing Assignment Guidelines* section of the manual.