# Lab # 2: Interfacing with Sensor device

Group # 08

MoulayLarbi ElAbidi, Raymond Hamilton, **Danae Moss**

Hand in date: March 17, 2017

Due Date: March 20, 2017

**Contributions:**

MoulayLarbi: Set up Galileo with YoctoLinux and wireless connectivity. Wrote code for Galileo and PIC16F18857.

Raymond: Designed schematic and block diagram for Galileo & PIC16F18857 interface

Danae: Wrote code for Galileo and PIC16F18857.

**Purpose:**

The purpose of this experiment was to enhance understanding of interfacing sensors and embedded systems, GPIO ports and bus protocols.

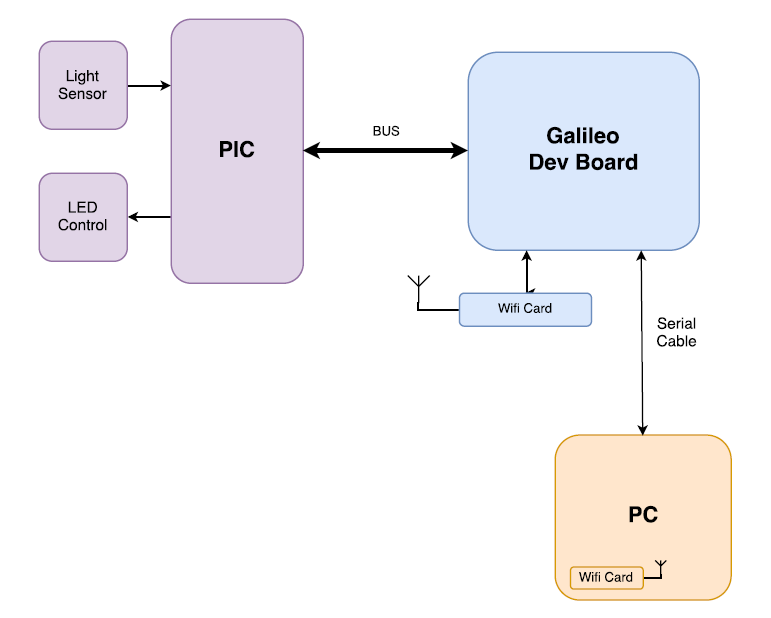
**Introduction:**

In this lab, we designed and constructed an interface between the sensor circuit from Lab 1 and a Galileo embedded computer. The completed device would have the Galileo accept and interpret a command input by a user. The main command would then cause the Galileo to send a request to the PIC, the PIC would respond with the 10 bit voltage reading from the photoresistor circuit, which the Galileo would then display to the user.

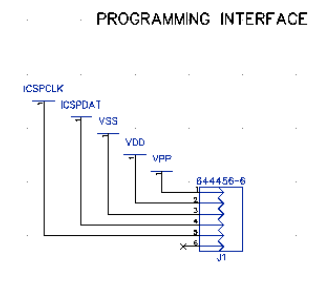
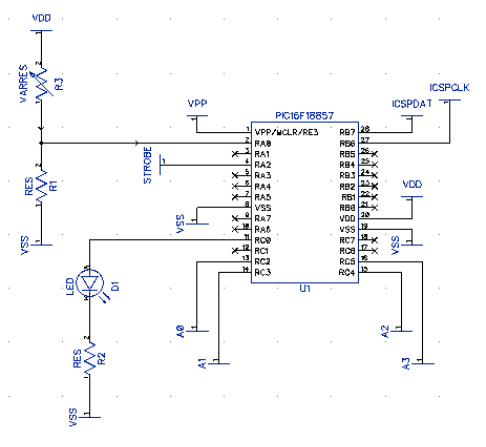
**Materials**:

* Fluke 45 Dual Display Multimeter
* PIC16F18857
* Intel Galileo Board Generation 2
* PIC-kit 3
* LED (used for testing and debugging purposes)
* Photoresistor
* 10kΩ resistor
* 100kΩ resistor

**Block Diagram:**



**Schematics:**

**Lab Methods & Procedure**

After successfully setting up the Galileo board and wireless configuration, we first attempted to familiarize ourselves with programming the interface by writing a basic use case code using the provided sample code as a framework. Without a reliable debugger for the PIC, we used the LED circuit from before while modifying the code to light the LED if code was producing desired result. After many iterative trials, I completed a functioning Galileo Write code for both the Galileo and the PIC. The algorithm is shown in the following flowchart diagram:

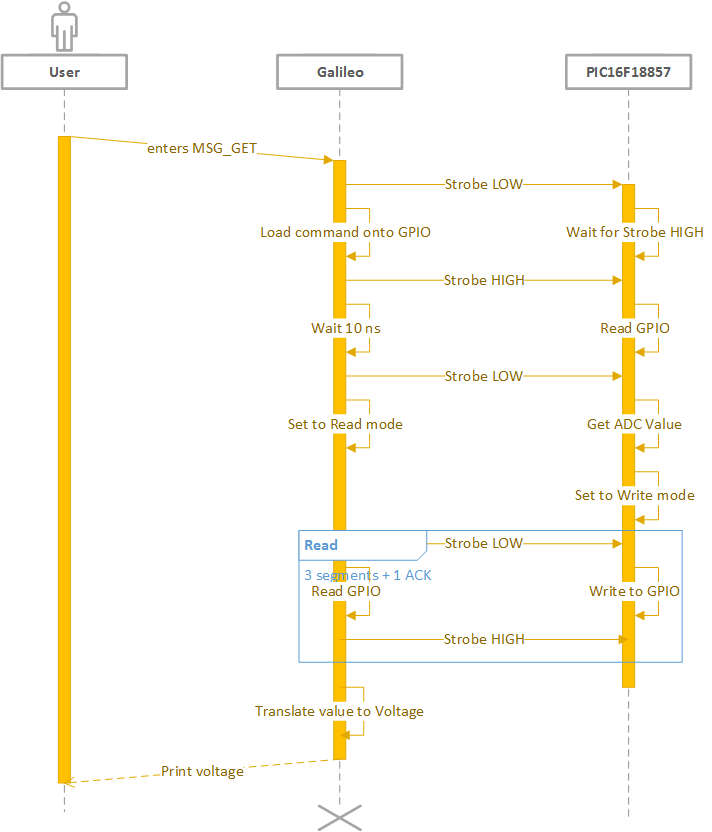
|  |  |
| --- | --- |
| PIC code flowchart | Galileo code flowchart |
|  |  |

After completing and successfully testing a Galileo Write / PIC Read exchange, I broke down the remaining tasks needed for this assignment into small testable components and formed a backlog loosely based on Agile Development strategy.

Backlog:

1. PIC – Add analog inputs
2. Galileo – Read mode: Can set GPIO and strobe to input
3. PIC – Write mode: Can set A0 – A3 and strobe pin to output
4. Galileo – Can write to console to prompt user for input
5. Galileo – Can Read 4 bits from GPIO pins
6. PIC – Can write 4 bits to A0 – A3
7. PIC – Can break down 10 bit value into 4 bit segments
8. Galileo – Can assemble received 4-bit messages into 10-bit value
9. PIC – Can translate analog voltage value to 10-bit value (import code from Lab 1)
10. Galileo – Can accept commands from user
11. Galileo – Can translate 10-bit value to voltage and print to console
12. Galileo – MSG\_GET command returns voltage from PIC
13. Galileo – Can perform Reset command
14. Galileo – Error Handling

The two functions combined would create a software timeline that would resemble this sequence diagram:



**Troubleshooting:**

When writing code based on the sample template, it was extremely difficult to tell what parts of the code were causing failures, since there was no reliable debugging system. Using the LED as a success indicator, I modified the code at multiple points, beginning with testing for individual pin values, then moving up levels of encapsulation to test logic and returned function values. Through this process, I was able to detect and correct errors until the Galileo could successfully use strobe protocol to write a four bit command to the PIC, and the PIC could interpret it.

The process for PIC writing to Galileo would be involving many of the same smaller pieces of logic rearranged, so I developed an Agile-style backlog to organize the coding process so it could be completed quickly. It was at this point I learned that my team had already completed the assigned code for both the PIC and the Galileo, and they required the lab kit to test their own code. Without access to the lab kit, I could not run or test my planned code design. In order to demonstrate my effort and understanding of the principles of this assignment, I included my software design and uploaded my first functional code to <https://github.com/rocket333d/Lab2Code.git> .

**Results:**

Galileo asks the user for an input command. If the user inputs the command for MSG\_GET, Galileo sends the command to the PIC using a strobe-based protocol. The PIC interprets this command and measures the voltage at the photoresistor port. The PIC breaks down the 10-bit voltage reading into 4-bit segments, then sends those segments to the Galileo using a strobe-based protocol. Galileo then reports this value to the console for the user to view.