Microavionics Final Project - Odometer

Carolyn Mason
ASEN 5519
December 13, 2015

I. Objective

The goal of this project is to design and program a bike odometer. A hall effect sensor will be used to calculate the users current speed, average speed, ride time, and total distance. A pulse sensor will be used to monitor the user's heart rate. The pulse sensor will be attached to the handle bars, so the user's pulse may be easily read. Lastly, a timer will start at the start of the ride to get the total ride time (including stops). An LCD screen will be used to display the data. Only one piece of data will be displayed at a time, so the user can toggle through the information by hitting a push button on the board. The three LED's on the board will be used to notify the user if their speed is at, above, or below their average speed. If the top LED is lit, the user is above their average speed, if the middle LED is lit they are at their average speed, and if the lowest LED is lit they are below their average speed.

II. Bill of Materials

Table 1: Bill of Materials

Item	Source	Part Number	Cost	Power Requirements	Interface Type
Latching Hall Effect Sensor	Sparkfun (datasheet: Hall US1881EUA.pdf)	US1881	$0.99	3.5V to 24V DC	Analog (I/O port)
Pulse Sensor	Sparkfun (datasheet:PulseSensorAmped GettingStartedGuide.pdf)	SEN-11574	$24.99	3V or 5V DC	Analog (I/O port)
Magnet	Own a magnet	-	Free	-	-

III. Hardware

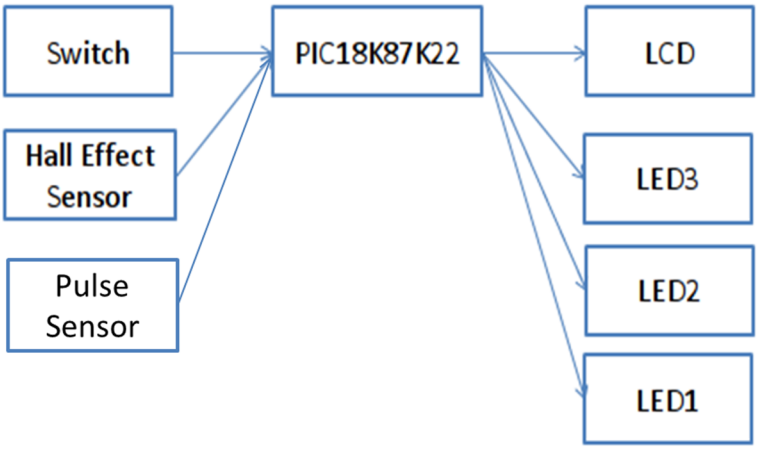

Figure 1: Simple hardware block diagram

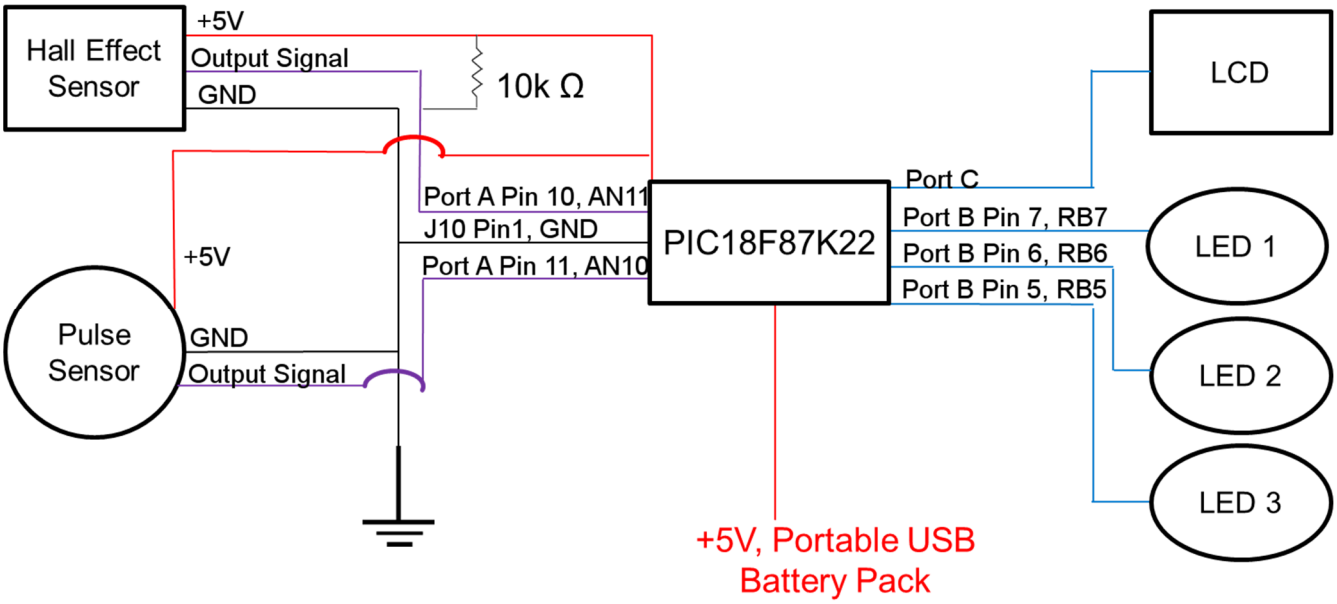

Figure 2: Detailed schematic wiring diagram with input (purple), output (blue), ground (black), and power (red)

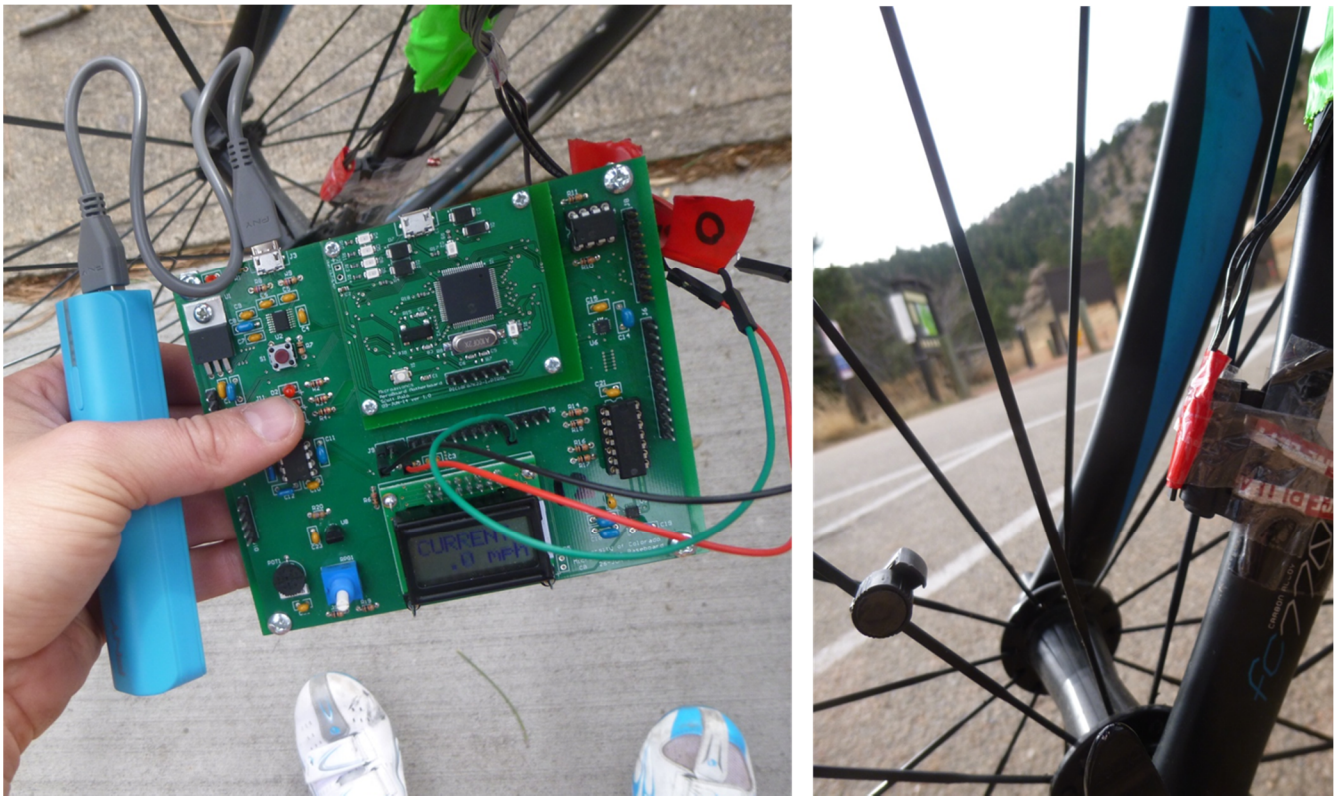

Figure 3: Odometer physical hardware and odometer placement

IV. Software

- Interrupts:
  The hall effect sensor and pulse sensor will be set on the low priority interrupt.

- Main:
  The main code is set up Port A for the Hall effect sensor and the pulse sensor, Port D for the switch, Port B for the three LEDs, and Port C for the LCD screen. The main code sets up and start running the timers. An interrupt will be used for the hall effect sensor. At the end of main the subroutine loop is called.

- Loop:
  The loop is called at the end of main. The loop handles the timing for the switch and updates the user with LCD stats on their current speed, average speed, max speed, ride time, total time, and total distance. There is a wait 0.1 second wait in this loop to prevent the LCD screen from updating too quickly.

- ISR_handler:
  The ISR handler checks if the flag was set and switch between checking the hall effect and pulse sensor subroutine. The handler then clears the flag.

- Hall_Effect_Sensor:
  The time between each pulse is monitored and the time is kept. This subroutine then calculates the current speed, average speed, max speed, ride time, total time, and total distance based on the count collected from the hall effect sensor. This subroutine also notifies the user of their current stats. The three LED's on the board notify the user if their speed is at, above, or below their average speed. If the top LED is lit, the user is above their average speed, if the middle LED is lit they are at their average speed, and if the lowest LED is lit they are below their average speed. At the end of the subroutine the current speed, average speed, max speed, ride time, total time, and total distance are all converted from unsigned integers to strings. These strings are saved and later displayed to the LCD screen.

- Pulse_Sensor:
  This subroutine reads the users pulse output. The pulse is calculated in the same fashion as the hall effect sensor, where the time between each beat is found. A vector of the last ten beat times is kept to find the average and calculate the number of beats per minute or bpm. These strings are saved and will later be displayed to the LCD screen.

- LCD:
  The display LCD subroutine handles displaying strings to the screen.

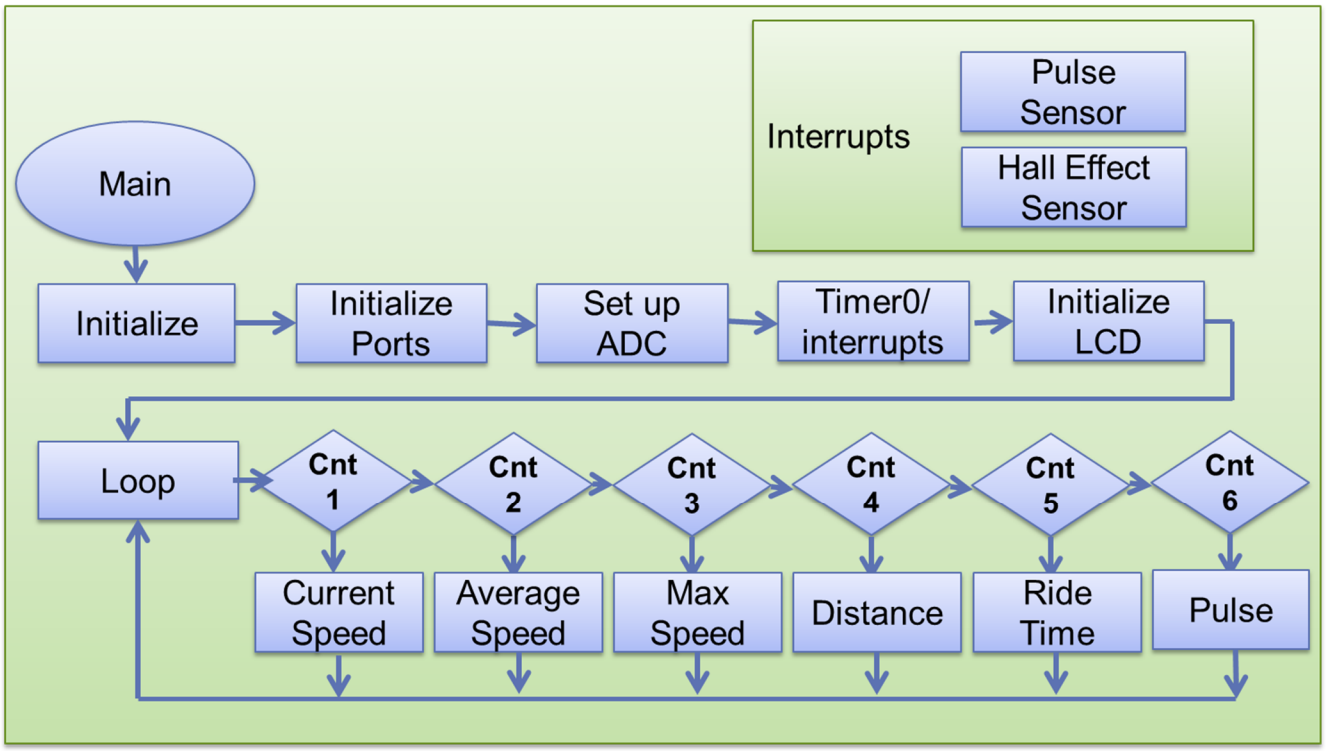

Figure 4: High level software diagram with main loop and interrupts

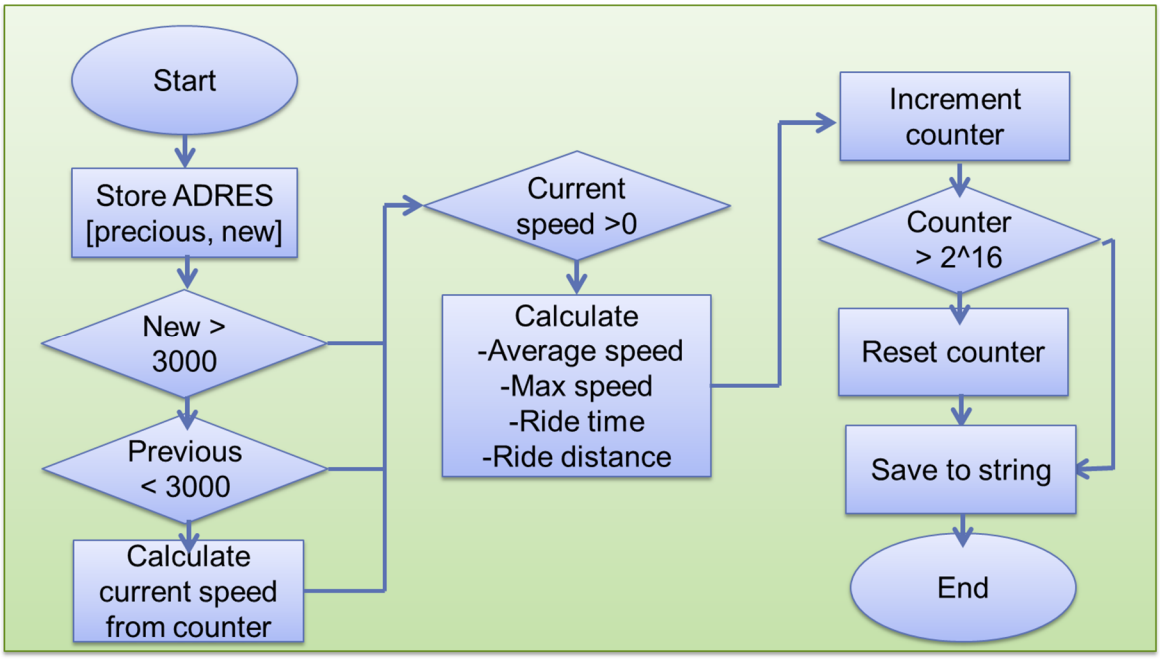

Figure 5: Hall effect sensor interrupt routine

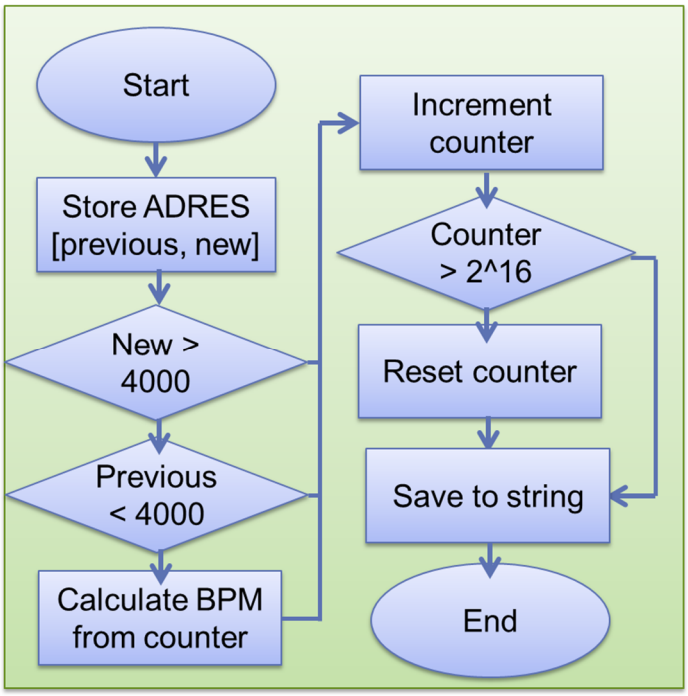

Figure 6: Pulse sensor interrupt routine

V. Approach

I first set up the code for the hall effect sensor. The first task was to set up the hall effect sensor on a bread board with a single LED. Here I learned the difference between latching and radiometric hall effect sensors and then proceeded to buy and test both types. A latching hall effect sensor will be set high when a magnet is brought close to it and only unlatch/ go low again when the other side of the magnet is brought close to the sensor. A radiometric hall effect sensor will be set high when the magnet is close and go low again as soon as the magnet is removed. Since timing proved difficult with the radiometric hall effect sensor, I chose to use the latching hall effect sensor and put two magnets on the wheel. The magnets were placed opposite of each other so that the hall effect sensor would be latched for half a rotation and unlatched for the second half of the rotation. This setup made finding the rotations easy to find and to track. This also made tracking higher speeds feasible. Instead of needing to recognize when the magnet passed, the software only needed to recognize half a wheel rotation.

The next step was setting up and testing the pulse sensor. This proved to be difficult because the base Arduino code for the pulse sensor did not work very well. I spent some time using the debugger and tracking the 'random' values. I later discovered that there were a timing limitation and the code structure supplied needed a much faster timing requirement. Instead of restricting my code, I used similar code to the hall affect sensor and tracked the time between each pulse. I tested the pulse sensor with the oscilloscope as well as output ADRES data using the UASRT in order to fully understand the timing and make sure I was capturing each beat. The Matlab plot is shown in Fig 7, below.

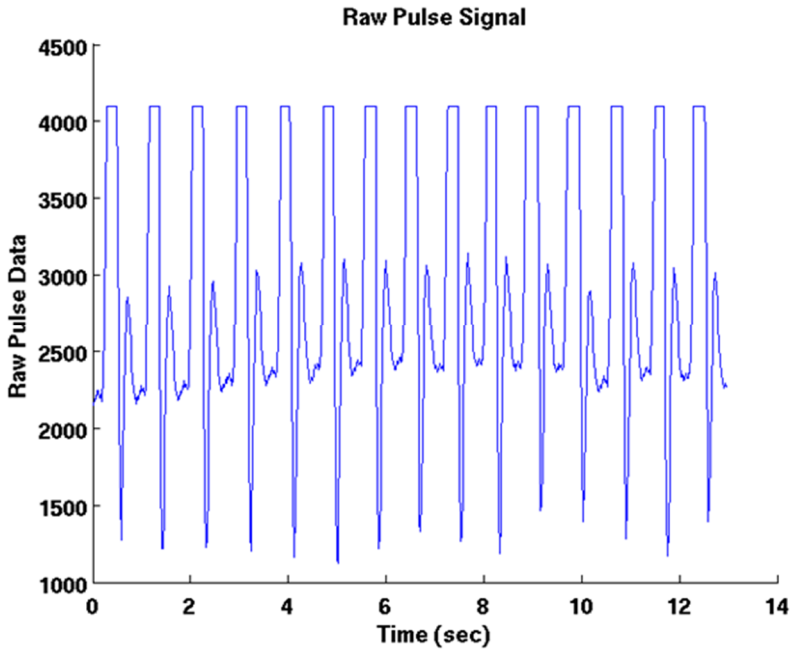

Figure 7: Pulse Sensor USART output then plotted in Matlab

After these sensors were tested and working on the bench I soldered them to longer wires so that the sensors could be attached to a moving bike. I also bought a portable 5v 1amp battery, so that the pic could be powered when biking. With the hardware setup I took the pic out on two separate rides. I found some issues during the first ride and made the necessary updates to support faster speeds and make the ride time and ride distance more accurate. Two pictures that I took while riding are shown below in Fig8 and Fig 9. The pulse shown on the PIC's LCD screen was 70bpm. The distance displayed on both the Odometer and PIC show 3.8 miles

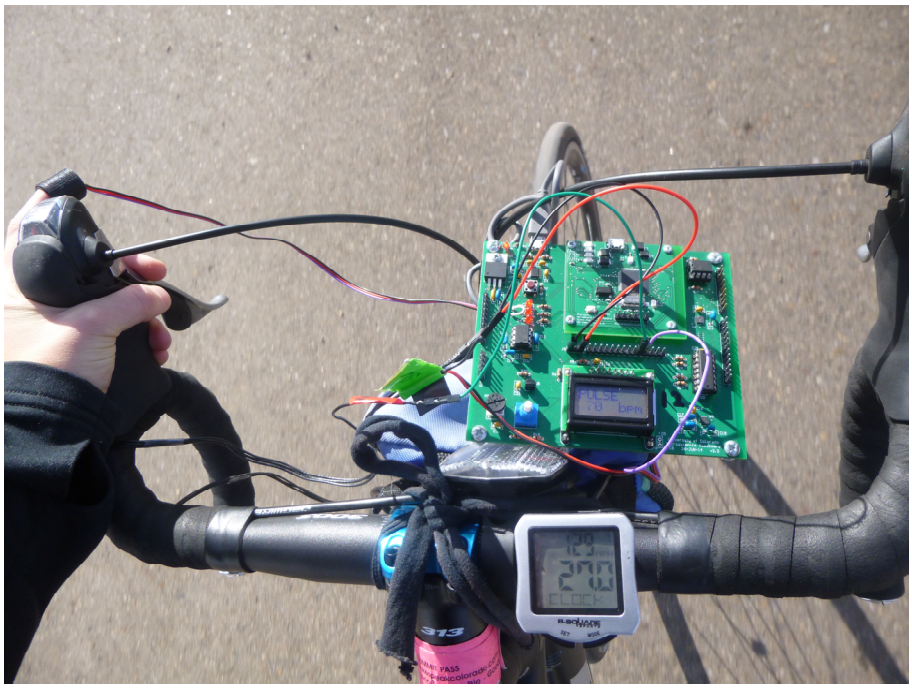

Figure 8: PIC set up while riding, with LCD screen and pulse sensor

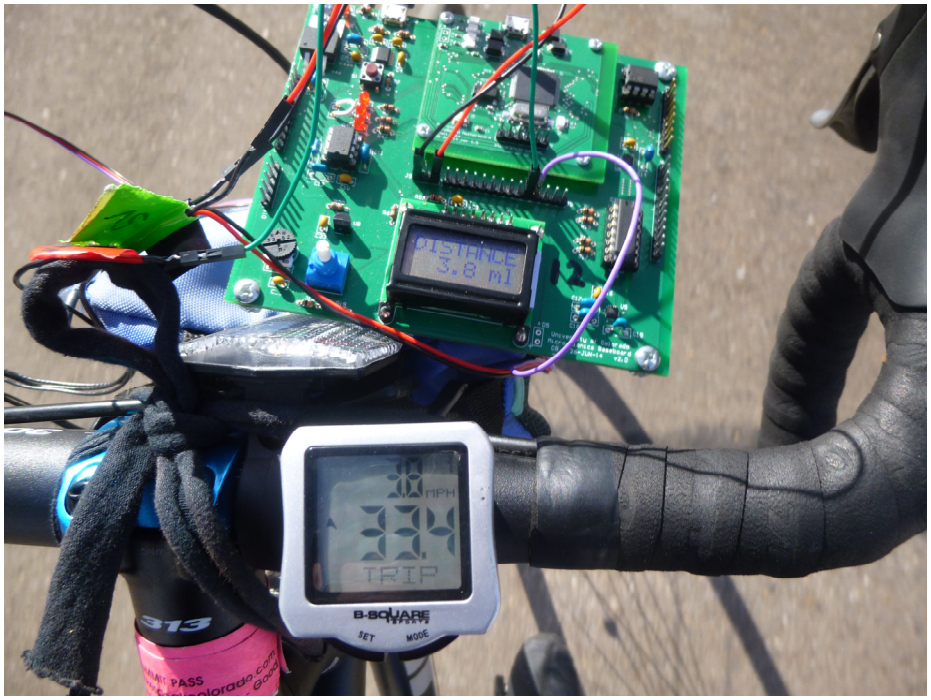

Figure 9: Photo of PIC showing ride distance

The last piece of the project was to set up each of the LCD displays. This was done in the main loop and relied on switch presses to change between display modes. Once the switch was set up this section of the code was easy to implement. The main consideration here was deciding how often to display new information to the user. If the loop was too fast the numbers changed too quickly and were hard to read, but if the timing was too slow information would be old and not useful. I chose to add a wait of 0.1second into the loop.

VI. Obstacles and Limitations

The three main obstacles with this project were the physical connections, portability, and calibrating the sensors due to timing resolution. To solve the issue of the physical connections, I used tape and solder. The hall effect sensor placement is important to read the magnets so tape was used to securely hold the sensor in place while biking. The second major issue with connections was solved with soldering the ground and +5v wires to the hall effect sensor's respective wires. This was necessary, since the pulse sensor did not work well off of 3.3 volts and there was only one 5v pin on the pic. Previous attempts at attaching the wires together were poor and soldering offered a solid connection which helped make the pulse sensor more portable.

In order to solve the power portability problem I purchased a portable battery (5v, 1amp). This battery proved sufficient to power the PIC and supplied enough current so the hall effect and pulse sensor could be powered from the same 5V line. The maximum amperage needed for the hall effect and pulse sensor is 31m amps, so the battery was more than sufficient.

The last obstacle was calibrating the sensors. The issue here was not fully understanding the timing within the code and what the pulses looked like. I did not realize that the pulse sensor has mini peaks

between the main peaks, which sometimes would get caught in the beats per minute solution. To solve this issue I uses an oscilloscope to view and measure peaks and the USART to plot data for easier viewing.

Since the code works by finding the time between peaks, there is a limitation that the biker cannot travel more than 60mph. If the rider hits this speed the hall effect sensor will not be read properly and the output will be poor.

VII. Conclusion

For this lab the hardware and software for a home-made odometer made and tested. The functionality of the odometer includes statistics that show the user's current speed, average speed, maximum speed, ride time, distance, and pulse. Each functionality of the odometer works and compares well with a store bought odometer. The odometer was tested both with bench tests and on two spate rides. The full limitations of the odometer have not been tested (60mph), but future tests can be run to verify these limitations. Future updates to the odometer could include using the accelerometer to know what percent grade you are climbing and also to save data to an SD card to have full ride statistics.

VIII. Appendix

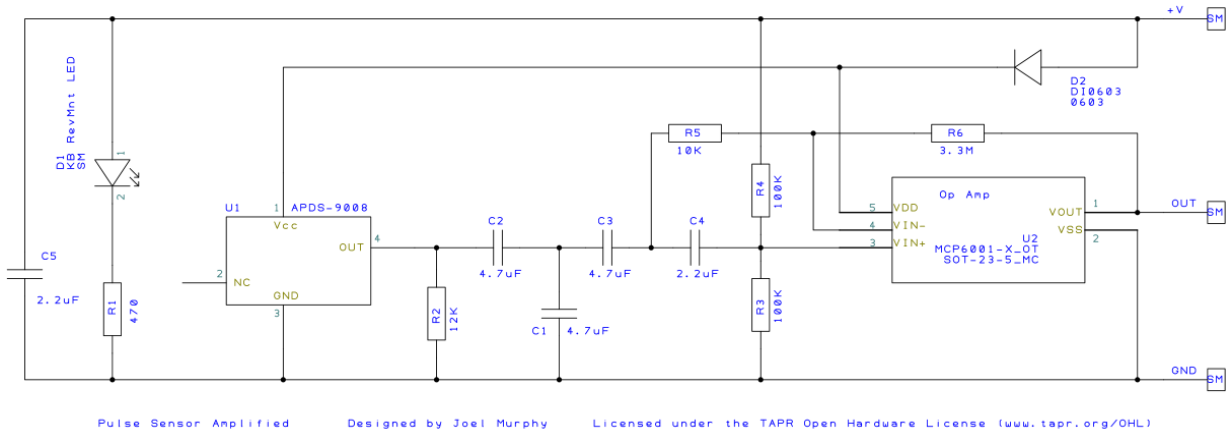

Figure 10: Pulse Sensor Schematic

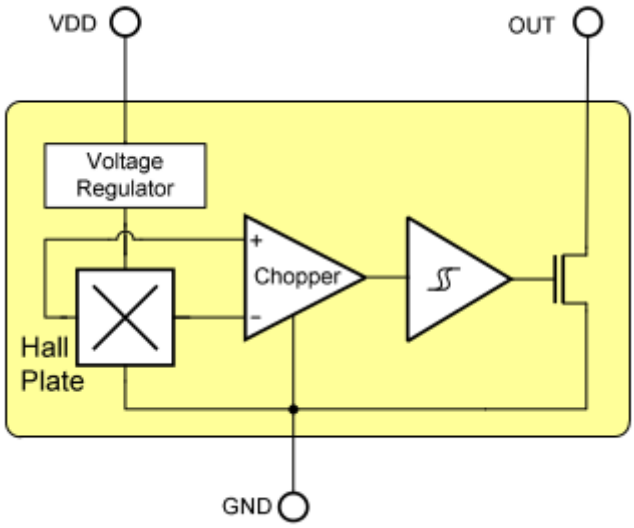

Figure 11: Hall Effect Sensor Schematic