# Feasibility Testing Report – Record EEPROM

*Team: P15001: Active Ankle Foot Orthotic*
*Engineer: Megan Ehrhart – Electrical Engineer*

*Test Date: October 10, 2014*

## Related System: Processing Data

This test looked at the feasibility of storing the gate information on the Arduino. By sorting the information in the EEPROM, the data would be saved on power down. However, it will take more time to read and write to EEPROM and therefore the timing of this procedure was very important.

## Testing Procedure

The first part of this testing was to look at where the information is going to be stored in memory. Three different pieces of information were stored. The first was the date that the EEPROM was cleared. This was to record when the information was initialized. The next two pieces of information were the researched average and the running or stored average that is updated throughout the program run time. The locations in memory are stored below:

***EEPROM Memory Mapping***

| Address | Data Stored | Reason |
|---|---|---|
| 0 | Year | |
| 1 | Month | Sore the date the EEPPROM was last cleared |
| 2 | Day | |
| 3 | Researched Gait Average | Average gait used for startup. |
| 4 | | |
| 5 | Stored Gait Average | Gait that is updated throughout the program |
| 6 | | |

## EEPROM Clearing

The next part of the test was to write code that would clear the EEPROM. This code was called P15001_EEPROMReadWrite. The variables that are written to memory are recorded at the top and called runtime variables. The date must be manually set here.

To run the code, upload it to the Arduino board and open the serial monitor. The board will then output what is stored in memory at that point. If the user wants to change the value of the memory location, they must type 'r' and press enter at this point. The EEPROM is then rewritten with the runtime variables.

## EEPROM in Code Execution

The last part of this test was to create a function that could write and read from the EEPROM during normal code execution. This was done in a class function that can be executed during normal Arduino Code.

### *Initialization*
The code will set the address that is used for the gait information that will be updated. In the example, memory address 5 was used. In addition to this, serial prints enable was set.

### *Read*
This function inside the EEPROM class will read the gait location starting at the initialized address. It will take the two bytes of this address and create the integer that represents the milliseconds of the gait cycle. The function has no inputs and will return an integer.

### *Write*
This function will write and integer value to the initialized address in the EEPROM. This will split the integer into the two bytes that can be stored in EEPROM. The function has an integer input and will not return anything.

## Results
The timing of this function is important. This was found using the Arduino's on board timer. This was also done without serial writes in order to make sure that the speed is representative. The timing is recorded below.

Read Function: < 1ms

Write Function: 4ms

Read Function and arithmetic operation and Write Function: 4ms

## Conclusions
It is clear from above that the write function timing is the limiting part of the functional of the EEPROM. However, because the current sampling time for the distance sensor is 50ms, this time would be not be so large a time that the EEPROM is not an appropriate solution. In fact, this test proves that the EEPROM is a good solution to storing gait data from run to run.

## Next Steps
Continue to use this code in testing and create more functions as needed.