Course: ME 549 – Microcontroller Interfacing (Bradley University)

Project: Play Day I

Author: M. Molter

Date: 28 JAN 2019

Desc: Class was canceled today, so we were asked to practice working with microcontrollers during the usual class period.

Objective: Start laying out potential course project and begin research.

Introduction:

While I have worked with the Arduino microcontroller extensively in the past, I have not had the opportunity to write custom sensor drivers/libraries in C++ or to program the Arduino microcontroller in ATMEL Studio. While these techniques make the Arduino a more powerful platform, work projects typically present the time constraints that prevent experimentation. Learning ATMEL Studio is not a good plan when the device *has to* work *today*.

I consider these two topics professional skills that I would like to develop during the course.

Potential Course Project:

Milestones:

In order to develop these skills, I plan on working towards the following milestones.

(1) Read a flow cell using an Arduino microcontroller.
(2) Write a function encapsulating the code to manage the flow cell in the background.
(3) Write a real, Arduino library for the flow cell in a separate file.
(4) Write an Arduino library for a serial device (e.g. electronic balance, linear photodiode array).
(5) Write a driver for a serial peripheral interface (SPI) device.
(6) Write a driver for an I2C device.
(7) Port the code from STEP (1) into ATMEL Studio.
(8) Write a driver for the flow cell as a separate library in ATMEL Studio.

Grading Criteria:

In order to evaluate my progress on the milestones above, I propose the following grading criteria/categories.

(1) Milestone completion
(2) Project documentation (e.g. reports, tutorials, and code commenting)

(3) Adherence to published code style guidelines
(4) Use of Git tracking/Maintenance of public GitHub page for drivers

Course Guest Lecture:

During the course, we are required to give an hour-long lecture on a topic of our choosing. Based on the timing of this lecture in the semester, I am planning on talking about "SPI Interfacing." I am planning on the lecture lining up with my completion of MILESTONE 5, writing a driver for an SPI device.

Notes:

Creating a GitHub Page:

I started by getting the GitHub page up and running. You can find the page at:

https://github.com/mmolter/ME549

I created the GitHub page using the GUI on the GitHub website, and then cloned it to my desktop using:

```
git clone https://github.com/mmolter/ME549
```

You can do the same if you want to see and edit all my code.

All commands in git start with the keyword `git` followed by a command keyword. When I create a new file and I want to add it to the repository, I use the `add` command followed by the name of the file. As is popular in UNIX operating systems, I can use an asterisk to add all the files in the directory. I can similarly use something like `*.pdf` to add all the PDFs present in the directory.

```
git add file1.txt

git add *

git add *.pdf
```

Whenever I make changes to code or files on my local repository, I need to commit (save). It is good practice to commit when you have reached a milestone in a coding session that results in again runnable code. For example, after adding a working function or completing a class module. The `-a` flag performs the add operation automatically, and the `-m` flag allows you to add a comment called a commit message.

```
git commit -a -m "completed the blink feature"
```

When you are done coding for the day, you push your changes up to the GitHub repository. For my page, I use my authorization credentials to make the change. The commands for these two operations are below:

```
git push
```

I created a `.gitignore` for the editable `*.docx` files from Microsoft Word. I plan to publish all my notes as `*.pdf`.

After initially adding the `*.docx` files to the repository, I had to learn how to remove a file from a git repository *without* deleting it from my local filesystem. This way, I can edit the files on my home computer, and only the finish `*.pdf` copies go public. The command is:

```
git rm --cached file.txt
```

I still need to work on choosing a license for the project and creating an index page for the GitHub repository.

Choosing Sensors:

The first sensor I will be working with is volumetric liquid flow sensor like the one in the image below. As liquid flows from the inlet port on the left to the outlet port on the right, it pushes a paddle wheel. The faster the flow of water, the faster the paddle wheel spins. Each time the paddle wheel completes a turn, the sensor outputs a pulse.

Often, the datasheet for the sensor has simple formula for converting the frequency of pulses to engineering units such as cubic meters per second.

$$Q = c_o \times f$$

These sensors typically have three wires: a +5 VDC supply, a common, and an output. Because the pulses can be on the order of 1-2 kHz, we will need to use a high-speed input (i.e. interrupt) to capture the signal