TATER October Monthly Status Report
10-31-2017

1 Description

This status report covers the progress and research conducted thus far for hardware targeting, algorithm analysis, and antenna design. Lastly, we discuss the areas of focus for the future.

Contents

2 Hardware Target

After receiving the shipment of parts, we prepared the board and began gathering capture data based on our previously defined tests. Our first goal was to determine if we could capture deterministic data.

We configured our setup to first trigger a reset on the target device by using an external microcontroller; which will then trigger the capture device. This allowed us to ensure we always got consistent results. Shown below is a bare-bones setup.

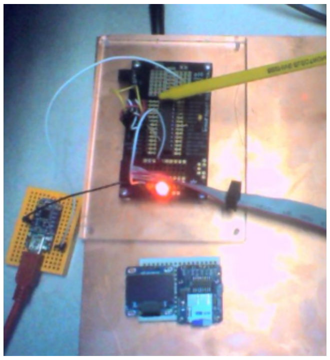
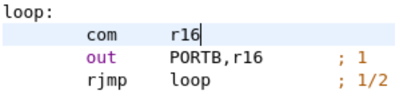


Figure 1: Bare Bones Setup

Based on our research, any I/O activity or off-chip accesses would have a higher detection rate. We wrote a small assembly loop to generate periodic I/O activity by complementing a register and pushing it to a port. (Figure 2). The comments illustrate the time in clocks for each instruction, where each clock cycle corresponds to 1us. Perhaps with a higher resolution device we would be able to see more data, especially in the microvolt range.



```
loop:
        com     r16
        out     PORTB,r16       ; 1
        rjmp    loop            ; 1/2
```

Figure 2: I/O Triggering Code

Using the automatically triggered oscilloscope, we obtained some nice results as shown in the below image. Here we can see the positive voltage rise as a change to all set bits, while a negative voltage rise corresponds to zeroed bits. We conducted similar tests by changing the amount of flipped bits, and noticed a linear correlation in the voltage change; making it an extremely deterministic.

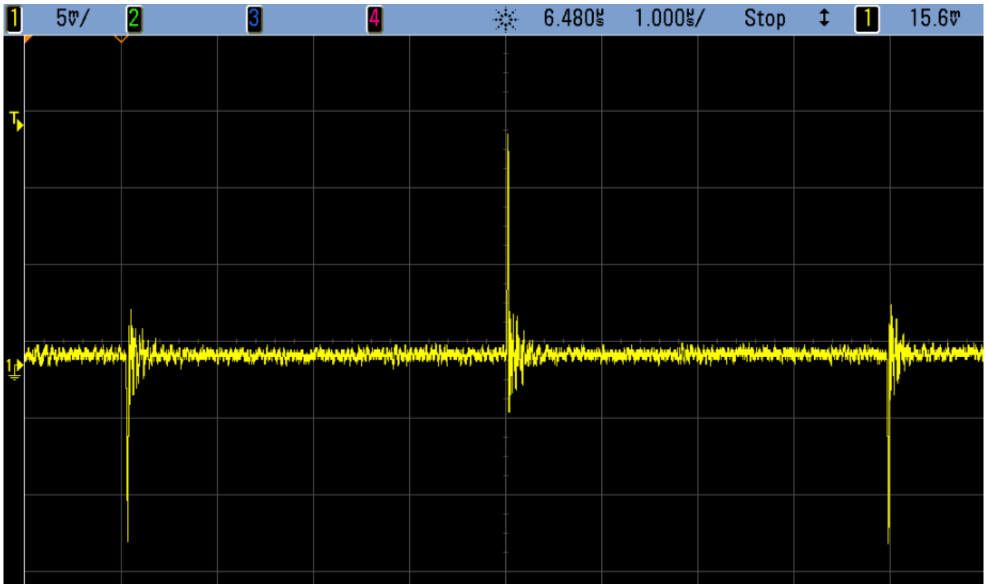
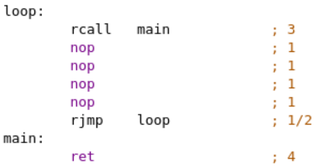


Figure 3: Waveform of I/O activity

We proceeded to test particular instructions that might be able to draw more current, in this case a pipeline flush in addition to a control change. This was accomplished by using the following snippet in Figure 4.



```
loop:
        rcall   main            ; 3
        nop                     ; 1
        nop                     ; 1
        nop                     ; 1
        nop                     ; 1
        rjmp    loop            ; 1/2
main:
        ret                     ; 4
```

Figure 4: Code for pipeline changes

We note that the response is extremely low; however we can still note small fluctuations around the region of interest in the `rcall` instruction, shown below in Figure 6 for 1us. Even with this small value, we feel that with a high enough sample rate we can notice these small fluctuations as well.

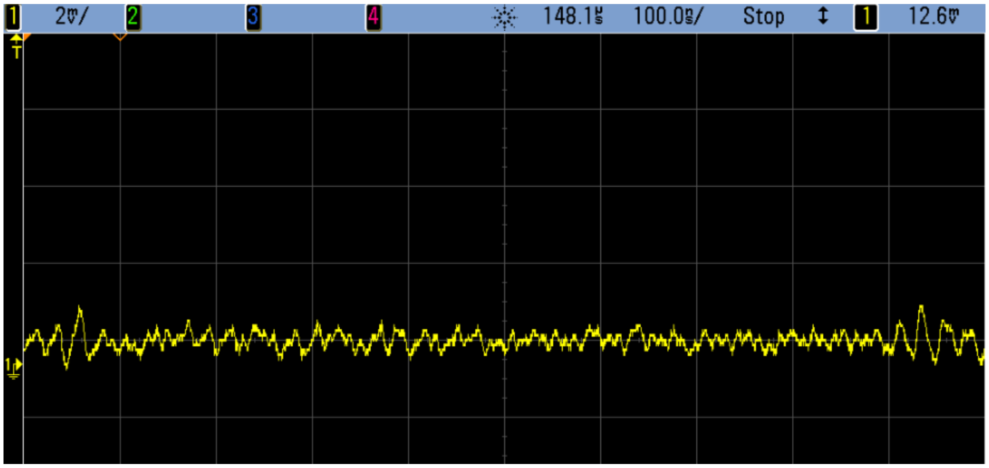
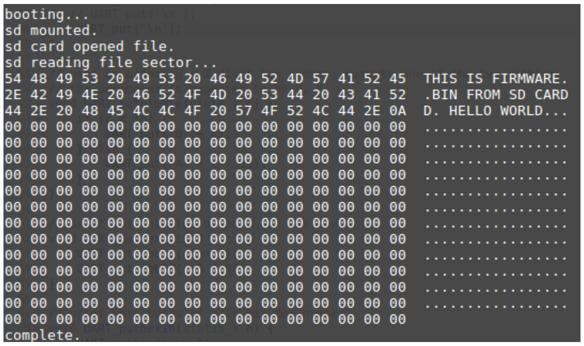


Figure 5: Waveform of control activity

Based on these results we wrote some of our initial bootcode using the Petit FAT File System Module. We are able to load a program off of an external SD card into memory and then proceed with execution.

We connected the microprocessor to a UART interface in order to have some debugging information; and added macro checking within the boot code so we can easily strip out this debug information in the future when we begin with training. Shown below is our target processor loading an external file from the SD card into memory. We feel that this method could also be useful for setting configuration options within the microprocessor, as that is one of our specified requirements, in addition to using an external LCD module to relay configuration back to the user for verification.



```
booting...
sd mounted.
sd card opened file.
sd reading file sector...
54 48 49 53 20 49 53 20 46 49 52 4D 57 41 52 45  THIS IS FIRMWARE.
2E 42 49 4E 20 46 52 4F 4D 20 53 44 20 43 41 52  .BIN FROM SD CARD
44 2E 20 48 45 4C 4C 4F 20 57 4F 52 4C 44 2E 0A  D. HELLO WORLD...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
complete.
```

Figure 6: UART interfacing with microprocessor boot code loading

The next section describes what we proceeded to do with this obtained information and data setup.

3 Algorithm Analysis

Based on our performed research on possible SVM (Support Vector Machine) analysis algorithms, we should be able to use this technique to analyze our data set. The goal of the SVM is to produce a model (based on the data) which predicts the target values of the test data given only the test data attributes. An SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. For this project's test data, the attributes are a series of time stamps and output values, where we will need to determine which set(s) in the series are "normal" or "not normal."

At first, we looked into the Random Forest and Decision Tree algorithms, but these will most likely not suit our data set because they are suited for multi-class problems and for mapping non linear relationships, where SVM is a good solution for us because it solves binary classification problems. We are also looking into LS-SVM which is a Large Scale SVM algorithm. This algorithm is still a possibility for us because LS-SVMs solve a massive system of linear equations (i.e. by using row and column incremental algorithms).

Shown below is the mapping of an SVM to a hyperplane, where each point represents a capture of our expected data. (Figure 7).

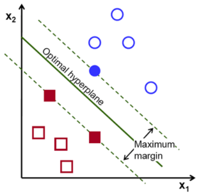


Figure 7: SVM hyperplane

As a trial run, we exported our periodic waveform data from the captured oscilloscope into a

CSV format for analysis on our computers. Using a short python script, we converted this to a LIBSVM format, a library and program for analysis. (We plan to attempt to use this library for testing and determining the best methods based on our input data once we have obtained more captures). Following this we then ran the LIBSVM training tools on our captured waveforms in order to build a model. This model consisted of about 4 million points. We then used the LIBSVM prediction tool with on our input data, and it delivered us with a 100% success rate. The goal of this was to determine if this tool could be useful in designing our later algorithm, as we need to perform a form of machine learning due to having requirements in the form of supporting configuration options, which will cause the waveform to never appear consistent, except in some cases.

The next section describes how we plan to implement an antenna with the best pickup for use in our training mechanisms.

4 Antenna Design

The design for our antenna primarily concerned how to properly shield the board and processor separately in order to obtain the highest possible gain. Based on our specified operating range of 1MHz, we note that in order to have a suitable bandwidth to capture these signals we will need a rather electrically small antenna.

To do this, we first use a metal container to enclose the board itself to prevent external EM radiation from leaking into the board components, as shown in Figure 9.

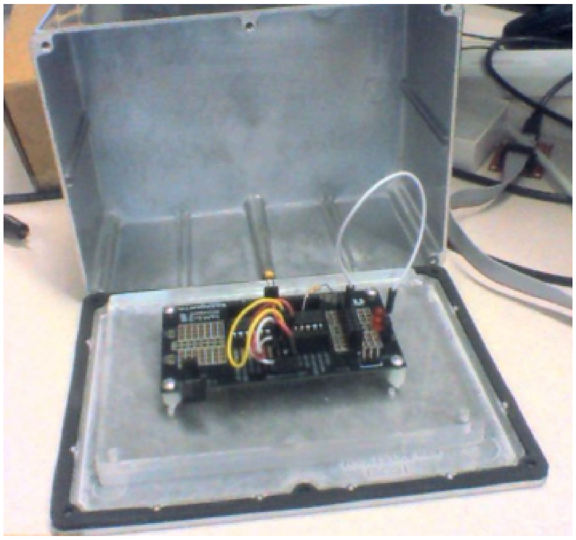


Figure 8: Shielding for EM spectrum

We then need to isolate the processor itself, which we plan to do by printing a 3D housing for the antenna. It will potentially be multi-layered, with a metal shielding and insulation from the outside world, with a first draft using low gauge magnetic wire to accomplish the task. Because the antenna may be required to be quite long, this allows us to wrap it and/or the PCB design within the housing itself.

Below is a simplified 3D model of the housing with an external access port which will then connect to coaxial cable and fed to our measuring device.

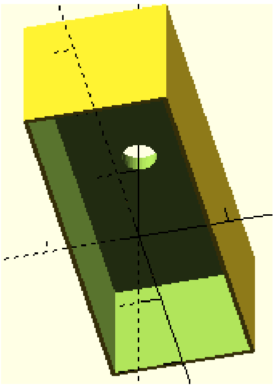


Figure 9: 3D modeled box

We have reserved the 3D printing room during this next week in order to speed development and begin testing of various antenna implementations.

5 Future plans

The next stages are to finalize our boot code implementation and hardware design, in order to ensure we can make repeatable measurements in order to begin training.

The crucial aspect of our design will be our method of algorithm and learning choice. We are still performing research for the best possible SVM algorithm, be it quadratic, linear, sigmodal or otherwise, and are meeting with Dr. Robert Heckendorn who will hopefully give us insight on the best routes to take in this regard.

Finally, we plan to 3D print our antenna shield and see which designs can result in the most pickup.

6 References

1. Support Vector Machine. *https://en.wikipedia.org/wiki/Support_vector_machine*
2. Large Scale Support Vector Machines and Theory. *https://pdfs.semanticscholar.org/975e/2e0204cb7a37f6b873795c425616a8678178.pdf*
3. LIBSVM. *https://www.csie.ntu.edu.tw/ cjlin/libsvm/*
4. Petit FAT File System Module. *http://elm-chan.org/fsw/ff/00index_p.html*
5. Meet the Challenge of Designing Electrically Small Anteneas. *http://www.mwrf.com/components/meet-challenge-designing-electrically-small-antennas*