5-2017

Target Detection with Neural Network Hardware

Hollis Bui
University of Tennessee, Knoxville, hbui2@vols.utk.edu

Garrett Massman
University of Tennessee, Knoxville, gmassman@vols.utk.edu

Nikolas Spangler
University of Tennessee, Knoxville, nspangle@vols.utk.edu

Jalen Tarvin
University of Tennessee, Knoxville, jtarvin@vols.utk.edu

Luke Bechtel
University of Tennessee, Knoxville, lbechtel@vols.utk.edu

See next page for additional authors

Author
Hollis Bui, Garrett Massman, Nikolas Spangler, Jalen Tarvin, Luke Bechtel, Kevin Dunn, and Shawn Bradford

Final Report

InTrueDeep

Date:	April 26, 2016
Team Number:	15
Team Members:	Luke Bechtel, Shawn Bradford, Hollis Bui, Kevin Dunn, Garrett Massman, Nick Spangler, Jalen Tarvin
Customer:	Dr. Catherine Schuman

Table of Contents

Executive Summary:

The primary goal of this project was to build a target recognition device using IBM's TrueNorth NS1e evaluation platform. The IBM development board provided to our team needed to be able to take input from an attached webcam, process each image frame to locate the target's position, and move a pair of motors to aim a Nerf gun at the target. The TrueNorth chip is configured with a pre-trained neural network built using the open source machine learning framework Caffe and IBM's custom Tea input and output layers. Furthermore, the launcher needed to be built from scratch, which required significant engineering design hours.

Unfortunately, due to challenges installing and configuring required software packages and tools, our project steadily fell behind schedule over the course of the semester. As of the end of April, we are unable to build a custom neural network using Caffe. However, we have been able to build a successful launching apparatus using a Raspberry Pi and Python. The launcher successfully aims and fires at a target with about 70% accuracy, although team members are still working to improve that number.

In January when the project implementation began, hardware was the team's main goal. Only one team member could work on installing Caffe from source and getting acquainted with the TrueNorth workflow. This was due to the fact we were only allowed to put IBM's export controlled software on a single EECS department computer, provided to us by Adam Disney. Once Caffe was installed, our team also had to learn how to setup GPIO on the Zynq microcontroller built onto the development board, which turned out to be a huge waste of time.

Requirements:

1. The system requires several hardware components.
    1.1. The hardware must include IBM's Neurosynaptic System 4[14] Evaluation Platform board (NS1e version revB).
    1.2. The hardware must include a Xilinx Zynq Z-7020 microcontroller, providing two ARM Cortex-A9 cores and configurable FPGA fabric.
    1.3. The hardware must include a TrueNorth neurosynaptic chip with 4096 cores with each core containing 256 neurons, 256 axons, and 65536 synapses.
    1.4. The hardware must include a webcam (Logitech C615) to gather real-time images.
    1.5. The hardware must include a pair of Vex motors and associated Vex robotic components.
    1.6. The hardware must include a Nerf Swarmfire launching device.
    1.7. The hardware must include a 3D printed rotatable platform on which to mount the launching device.
2. The system must include several software components:
    2.1. Zynq firmware must be able to coordinate inputs and outputs.
    2.2. The software must use Caffe defined training algorithms to build a neural network for configuring the TrueNorth chip.
    2.3. The chip must cease to be in learning mode once initially programmed.

3. The system must be able to collect images.
    3.1. At least 25,000 images will need to be gathered for training, testing and verification sets.
    3.2. A real-time webcam feed will deliver input images to the Zynq chip.
    3.3. Each image will be at least 720 x 480 pixels in resolution.
4. The neural network must be able to identify a target within an image.
    4.1. The off-chip classification accuracy must be at least 95%.
    4.2. The on-chip classification accuracy must be at least 80%.
    4.3. The first target will be a red circle. As our classification accuracy improves, more complicated targets such as college mascots will be tested.
5. The TrueNorth and Zynq chips must be able to exchange information.
    5.1. Serial communication between the board and a host computer must be setup to verify classification accuracies between on-chip and off-chip neural networks.
    5.2. The TrueNorth chip must take an image file on input.
    5.3. The TrueNorth chip must output a series of bits encoding the image's classification information.
6. The Zynq chip must be able to move a launcher towards a target.
    6.1. Coordinates sent from a host computer will be able to move to the precision of the motors.
    6.2. The Zynq chip must be able to decode coordinate information from the TrueNorth chip's output.
7. The system components must be assembled into a single product.

7.1. Classification accuracies will be gathered from a series of tests.

Change Log:

On Sunday, 24 April 2016, our team decided to focus on reliably getting the launcher to work with a Raspberry Pi and a laptop emulating TrueNorth. This is not to say TrueNorth is incapable of properly classifying the images desirably, just that our team was unable to meet the initial requirements given the one semester time frame.

Documentation of the Design Process:

The design process our team followed began with a division of labor between hardware and software tasks. Our team is very representative of the EECS department: four computer science majors, two electrical engineering majors and one computer engineering major. The first main hardware task was designing the launching apparatus. After many hours of discussing the system requirements and several CAD model revisions later, our team arrived at a unique solution that combined 3D printed components with Vex motors to mount and move the Nerf gun.

On the software side, a dataset needed to be built before Caffe training could be started. Our team also needed to decide how training and testing data was to be labeled for classification purposes. We decided it would be easiest to break each 1280x720 RGB image from the USB web camera into a collection of one thousand six hundred 576 pixel tiles. This allowed us to treat each image as a 40x40 grid of 18x32 subimages. Each tile could then be checked to see if it contains enough red to potentially be a part of the target.

This basic algorithm was first implemented in Python using OpenCV on a very small collection of images similar to Figure 1 below. Our first generation of software was able to find the center of the target as well as its borders, but the software's output needed to be verified for correctness before more progress could be made. Also, for Caffe training purposes, our output needed to be in the form of an lmbd (Lightning Memory DataBase). Because Caffe installation had not finished by this point in the semester, progress was halted for several weeks.

Figure 1: Kevin and Hollis generating an early dataset

Installing Caffe according to IBM's specifications is only possible for those brave souls willing to dive into the world of package managers and makefiles. The HP ZBook given to our team by Adam Disney originally had CentOS 6 installed when Adam took it to San Jose for

IBM's Neurosynaptic Bootcamp last summer. Even after several weeks of training, Adam was never able to fully install Caffe on the machine himself.

Before the laptop was given to us, Adam kindly reinstalled the operating system on the machine in hopes of easing the task of installing Caffe from source. This time, Fedora 23 was chosen as the host OS. Some of the Caffe and IBM installation prerequisites include Nvidia's Cuda toolchain, Google's protobuf, gflags and glogs libraries, and OpenCV version 2.4, among others. The first time Cuda drivers were installed on the machine, the screen went blank and the machine's operator was stuck using a command line only interface.

After tty'ing into the machine for several days in a row, the Cuda drivers and toolkit finally got installed. After that, it was simply a matter of using dnf, Fedora 23's default package manager, to install the remaining libraries and tools. Unfortunately, OpenCV needed to be compiled from source, but the make process was relatively straightforward since it used cmake. The next challenge came in the form of properly configuring Caffe's compile process, which also used cmake.

Fedora 23's default compiler is gcc 5.3, which was more recent than IBM's installation instructions recommended. Indeed, when trying to compile Caffe with gcc 5.x, the generated libraries and executables failed to link with the precompiled OpenCV and protobuf libraries. To overcome these problems, gcc 4.7 was installed on the machine. This new compiler would be used for installing Caffe. It was clear that packages installed from source would have to be recompiled, which took another day to accomplish.

By the time the Caffe cmake process was reinitialized, failures kept occurring at the linking stage. As it turns out, the application binary interface for programs compiled on Fedora

23 is fundamentally incompatible with older versions of gcc. The root of the issue was gcc 5.x being the system default compiler. Thus the easiest solution, even after several weeks of fighting to get a Caffe compile setup, was to reinstall the operating system with an older default gcc version.

An attempt was made to install CentOS 6 on the laptop, but after several attempts, the installations kept failing. Our team then consulted with Adam on the issue, and he advised installing an older version of Fedora. Eventually the computer was successfully reimaged using Fedora 21 with a default gcc version of 4.8. The next step was to follow through IBM's installation guide again, meaning all of the previous steps had to be repeated on this new operating system.

Although some hiccups did occur during installation on this machine, the build process ran much smoother than before. For instance, we could no longer use Cuda 7.5, which requires at least gcc 5.0 to work, but Cuda 7.0 installed successfully. OpenCV and protobuf both needed to be installed from source, but the procedures were the same as before, so they simply amounted to reconfiguring some cmake variables and recompiling.

The first time the Caffe build completed, there were still some missing components that were never made due to an improper cmake configuration. Both pycaffe and matcaffe failed to build for their own reasons. After downgrading OpenCV from 3.0 to 2.4 and reconfiguring some matcaffe installation settings, both packages finally built and the team member working on the project could breath a sigh of relief. For the moment.

By the time Caffe was fully installed, spring break had arrived and the team disbanded for just under a week. Most of the team members were willing to meet the first Saturday of the

break and the Sunday before classes resumed, so progress on the hardware side of things continued to move forward.

In order to meet our project goals, we would require all software and hardware to run exclusively from the TrueNorth development board. This was becoming more and more of a challenge, and the team decided as a whole it would be best to begin testing pulse-width modulation motor controls using a readily available Raspberry Pi. This was an alternative to using the Digilent Zybo board purchased for the project, and was probably the better choice. Using the Zybo would mean setting up a Xilinx toolchain using Petalinux, similar to the workflow required to communicate with the Zynq chip built onto the TrueNorth board.

There were several difficulties associated with getting GPIO working on the TrueNorth board. Firstly, there was zero documentation on the board other than a set of bootcamp slides and a condensed two page datasheet, both provided by Adam. After posting to the IBM forum, we found the proper Samtec connector part number, but no indication of a breakout board solution was provided. Luckily, the company Proto Advantage provides a suitable breakout board. Once the connector and breakout board arrived, they were soldered together and connected to the TrueNorth's GPIO connector.

To test the GPIO ports, a simple Python program was written that could be run while ssh'd into TrueNorth's natively running Ubuntu. The program simply sourced all available GPIO ports and echo'd 1 into each pin's value file after setting the pin's mode to output. Using an oscilloscope, we probed each of the GPIO pins, but no voltage signals were seen. After an afternoon of debugging without progress, it was the team's consensus that the only way to get

GPIO working on this device was by writing a kernel module and inserting it into the Ubuntu kernel at runtime.

At this point, we were uncertain at best how to proceed. The team submitted another post to the IBM forum asking for guidance on dealing with the GPIO. Unfortunately, the response wasn't very helpful. Whoever replied said IBM's team of engineers didn't have a solid setup for interfacing with the board's hardware or firmware, and instead promoted us to explore TrueNorth's capabilities from within Matlab.

This didn't help our situation in the slightest. We left one team member to learn to use Xilinx's Petalinux tool to build a kernel module. Unfortunately, without the Ubuntu build setup used to get TrueNorth into its current state, it appears impossible to build a generic kernel module given no device tree configuration or raw boot image. During this time frame, the HP laptop was being used for sshing directly into the board, so progress on the Caffe side of things was forced to halt.

Although it felt like very little progress was being made, Kevin and Luke were successfully operating the launcher using a Raspberry Pi's GPIO ports. Garrett transitioned out of his role dealing with Zynq hardware into actually using Caffe. In order to program TrueNorth as per IBM's specifications, a neural network needed to be built using the provided Tea layer patch. However, complications arose before this task could be completed.

A second video was recorded with images like those in Figure 2. This video was about twelve minutes long and featured the target moving over the entire range of the camera's field of vision. In order to turn the raw video into labeled input for training, old Python code was rewritten to label each tile inside each frame with a 0 if it contains less than 50 red pixels or 1

otherwise. The tiles were then saved as .jpg's in their own directory, and all labels were written to a file. The data was then passed into Caffe's convert_imageset function to generate an intermediate lmdb before the data could be used for TrueNorth training.

Figure 2: Example image for real test and train datasets

In order to get our Caffe lmdb into a TrueNorth lmdb, IBM recommended we use one of their precompiled programs called tn-signal-processor. This program at first seemed to have a bunch of signal processing utilities, such as random affine transformations and filters. However, it failed at its most basic task: converting a plain Caffe compatible lmdb into a Tea layer tn-lmdb. The program would always find the first image in the database and then either quit or hang forever. At one point, a tiny dataset of about eighty images was created to test tn-signal-processor, and the program failed to quit after running for two days.

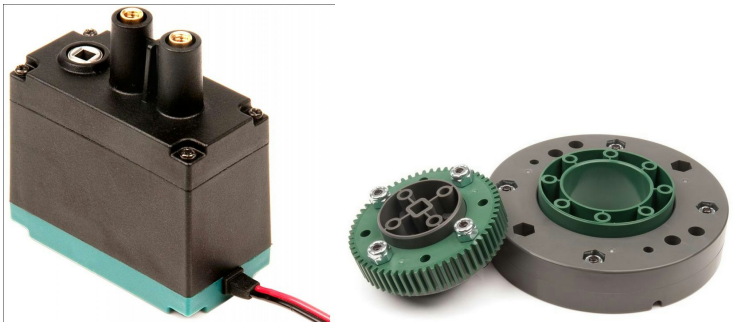
Several attempts were made to modify the lmdb generating code, and different image formats were tried as well. It appears as though the lmdb offsets internally aren't being incremented properly in tn-signal-processor, but the only way to know for sure what goes on inside the program is by reading the code. Unfortunately, Adam told us IBM wouldn't be willing to give us their source code. It seems we were indefinitely stuck up a creek without a paddle, canoe, lifeboat, lantern, or even reliable documentation.

On the hardware side, the main design parameters were to provide physical outputs for the software that would allow activation, such as pitch rotation, yaw rotation, and firing actuation. The team first decomposed the actuations and took an in depth look at the requirements for each output and the interfaces that would have to be used.

To begin, the team contemplated the yaw motion. The yaw motion would be responsible for aligning the launching apparatus in the XY plane. Since, the launching apparatus had to be repeatable, and its position absolute, a simple motor could not be used for this motion. The motor would have to have an integrated encoder that would provide real time feedback of its position. Another option would be to use a servo. A servo can provide high accuracy angular motion and is often signaled with a word that dictates its desired position. The second requirement for the yaw component is to be small, to provide easy integration to our assembly, and to provide enough torque to rotate the upper assembly and launching apparatus.

With this in mind the students began looking at low cost hobby motors and servos. The students found several options but VEX Robotics inc. provided a large variety of solutions to meet our needs. VEX not only provided small affordable motors and servos but also provided support systems, interfacing components, and large amounts of technical information for using

their products. From VEX's product list the team found a perfect match for the yaw motion. Vex offered a small 2 wire 393 motor that could be fitted with an encoder to provide position feedback. Secondly, they provided a bearing kit, which could be used to support the upper launching assembly, and the two wire motor would mate perfectly to the bearing kit. This was an excellent start for the yaw motion.



Left: *Vex 393 motor* ***Right:*** *Vex turntable bearing kit*
Image Curtosiy of www.vexrobotics.com

Next, the team began looking at the pitch motion. The pitch motion would be responsible for moving the launching device up and down in the Z plane. This component would also have to be highly accurate, small, and provide sufficient torque. Once again, the team turned to VEX. The two wire motor was sufficient for yaw motion but could not be used for pitch. The two wire motor could not be used to hold its position against the force of gravity pulling the launching mechanism back down. However, it would provide sufficient feedback to ensure the correct position had been attained. The team then contemplated a solution to finding a new component or modifying the design to somehow hold the launching apparatus in place. Two solutions were presented. The first was to attach the motor to the base of the table, instead of to the axis or rotation, and use it as a winch. The winch would apply force to a chain and turn the pitch axis via a gear. This would allow for a torque advantage at the motor. If the torque advantage was great

enough the motors drag may be sufficient to hold the launching apparatus in the correct position. The second alternative was to use the three wire servo from VEX. As stated above, the servo could be used to turn the axis to the exact position. Also, since servos are signaled to an absolute position they don't de-energize when arriving. This meant that the force of the servo would be applied continuously to hold it in position. However, the servo was limited to 100 degrees of rotation. The team felt that the servo would be the best choice to provide pitch motion. Even though the servo has a limited range of motion its ability to hold the location made it the best choice for this application.

Vex 393 Servo
Image Curtosiy of www.vexrobotics.com

The third actuator to be considered was the launching apparatus. Several solutions for launching were considered: trebuchet, airsoft guns, and a Nerf gun. A trebuchet is a medieval catapult type device that used gravity to hurtle object through the air. This was obviously not a viable solution due to the weight, size, and accuracy. Next an airsoft gun was considered. Airsoft guns can fire small plastic or paint pellets at high velocity towards targets. The airsoft gun can also be loaded with multiple projectiles and the force behind the projectiles can be compressed gas. That would mean that the gun could be fired multiple times during testing without having to be removed or reloaded. The airsoft gun's ability to repeatedly and consistently fire made it an

acceptable choice. The last solution was a Nerf gun. Nerf guns are less consistent than airsoft guns but the concern for injury during testing is significantly reduced. Also they are made out of plastic, therefore they are often lightweight and inexpensive. The repeatability of the Nerf gun was a major concern. Most of the models the team found required a mechanical action to be recocked after each shot. After further investigation the team found the Nerf Swarmfire could hold multiple projectiles, had an electronic launching mechanism, and the trigger was electronically energized. This was the best solution due to its ability to be easily interfaced with the I/O for firing.

Nerf Swarmfire
Image Curtosiy of http://nerf.hasbro.com/en-us

The solutions above were chosen to fulfill the project's requirements. However further investigation needed to be performed to ensure integration of all the components could be achieved. The most important integration is signal and power requirements. The components would have to be powered by an external source. Vex components were designed for hobbyist and would integrate easily with the project. Both the Vex servo, motor, and encoder would accept and transmit signals at a voltage that is safe for the Digilent Zybo board. Secondly, the Vex components all operate at the same voltage for the necessary drive power. Using the specifications from Vex's website the team found an acceptable power supply that would output 5V and supply enough current to meet the needs of the project.

The last component to be integrated was the Nerf gun. Our model's voltage requirement was 9V. However, since the gun came with its own power supply "battery pack", only the triggering mechanism had to be integrated. The triggering mechanism was a simple mechanical contact. This contact open or closed the circuit and provided power to the launching mechanism. To recreate this contact which could be controlled by the I/O the team decided to use a relay. The relay could be energized by any GPIO pin and the contact for the gun could be made. This would allow for the gun's power supply to be completely isolated from the driver board. Those are the solutions to integrating the I/O for the project.

Once all the components were chosen for the project a structural support system had to be created to securely mate all the components. The team would have to create a strong but light custom structure to support all the launching and aiming components. There were several solutions such as wood, metal, and 3D-printed plastics. Of these, metal would be heavy and difficult to customize. Wood is lighter, but would be bulky and hard to attach the Vex components. Plastic was a viable solution because plastic can be light, strong, and easily customizable through the use of a 3D printer. Now that we had a solution, the design of the support structure had to be modeled in software that would create a printable file. We turned to Autodesk's AutoCAD Inventor. Through the use of this program, we were able to create their design and provide perfect mounting surfaces for the Vex components. 3D printing the structure also allowed the components to be small, yet maintain strength by increasing the density of the print in the support volume. This solution proved to be the best and most economical choice for the support structure.

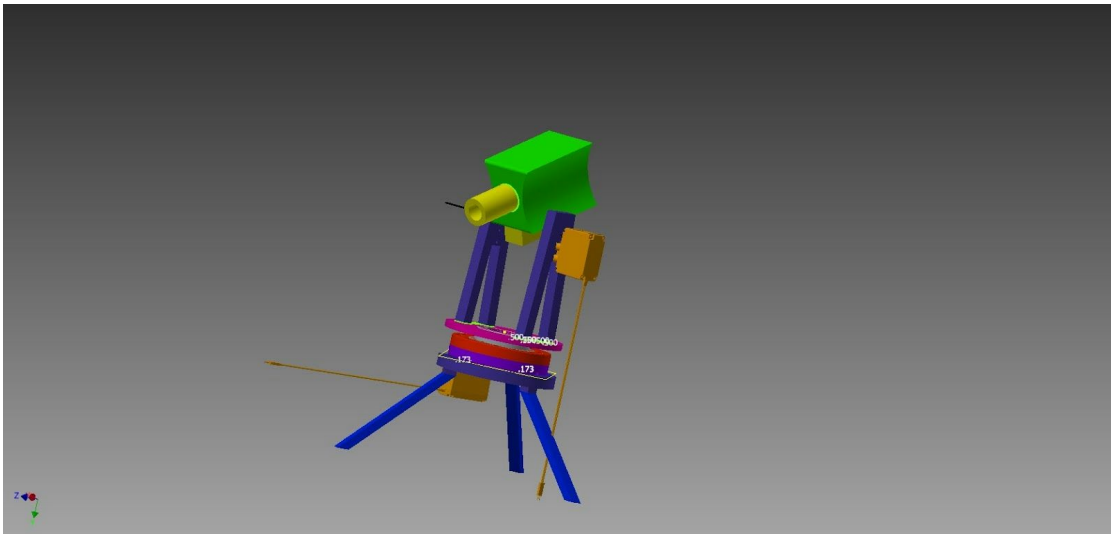
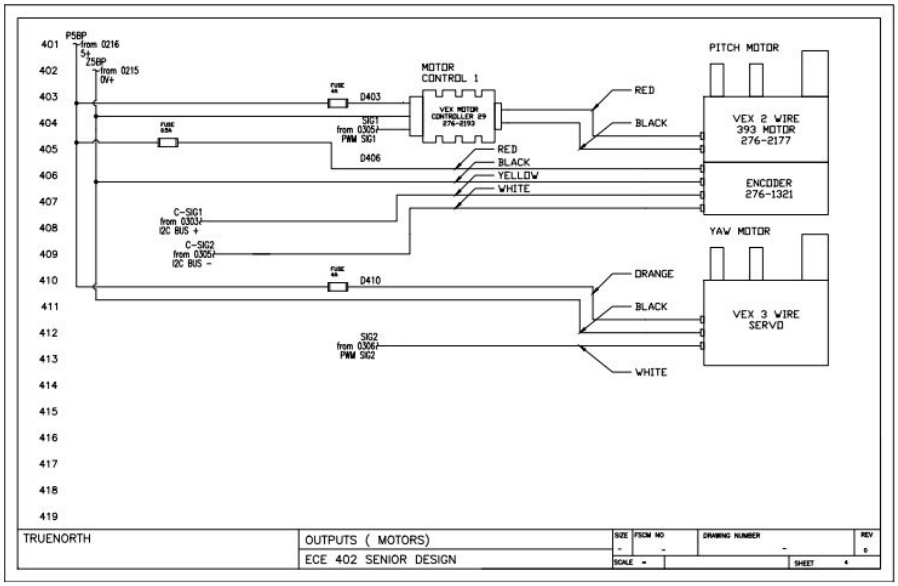
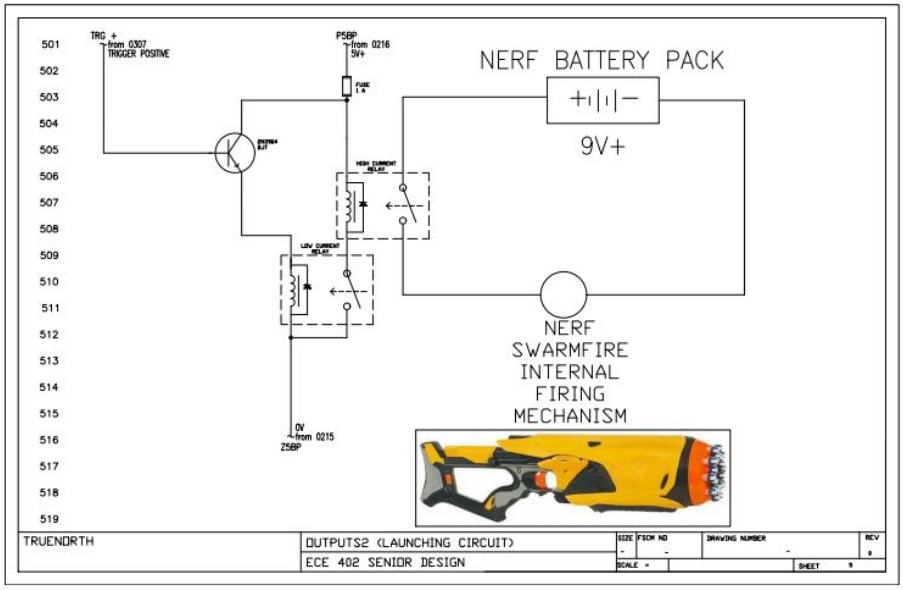


Image of full assembly modeled in AutoCAD Inventor

Once the components were purchased and the structure was modeled the team began a plan to implement the design. Part of the team worked on 3D printing the required parts and ensuring that the tolerances for the parts matched the model. The other part of the team worked on creating a hardware drawing that would serve as a wiring guide. Once again Autodesk software was chose to create the wiring guide. Using Autodesk AutoCAD Electrical, detailed guides were made, through the creation of the hardware diagrams components such as fuses, wires, and connectors would be discovered. This is a design step but is often performed after design because certain parts and interactions cannot be easily foreseen. These parts were ordered and upon arrival assembly began. The team's diligence in creating detailed models lead to a smooth assembly and a quick transition from implementation to testing.





A few Images of the detailed hardware drawings created in AutoCAD Electrical. Details include wire color, fuse requirements, part details, available voltage, etc.

Lessons Learned:

Unfortunately, our project taught us a lesson in working in parallel versus working with delays: while each member of the team had an idealized job (Hollis had wanted to work with the neural network itself, while Shawn would work with embedded software, for example) roles changed as the team fought to overcome delays that crippled progress until certain bottlenecks were passed.

On the hardware side of the project, our team progressed at a steady pace, but only after the 3D-printing was finished. This delay snowballed into forcing the team to make up for this time during the middle of the semester, which may have contributed to taking focus away from other projects during this time. In the future, it may have been better to have started construction as soon as the design was finalized, which was quite early in our development; overall, while the material construction was a success, it could have been completed earlier, with many of us lamenting how we did not take advantage of Christmas break for parts production or ordering.

On the software side, the major problem of our project was unfortunately what it focused on: the TrueNorth interface. As discussed between team members and to our customer, the documentation for the TrueNorth board was difficult to traverse. The compilation of Caffe, which was an integral bottleneck for programming on the board, took significantly longer than expected due to compilation errors related to the poor documentation. In addition, the original idea of working in parallel with multiple programmers accessing the board remotely never came to fruition, and as the semester neared its end -- and Caffe was finally compiled -- the team became too occupied by other tasks to be able to help Garrett with Caffe.

From a managerial perspective, this project revealed quite a few instances where resources were simply allocated poorly. The primary example of this is with our team going over budget. We ordered a $200 Zybo board that was never used, although it is a very useful development board for the Zynq microcontroller platform. This effectively reduced our budget by about 40%, forcing team members to shoulder the costs of electronics needing to be purchased later in the semester. There were also several wrong parts that were ordered, so time was wasted having to wait for the correct parts to be delivered.

A project is more than managing money however. Our team contains plenty of great talent, but getting all schedules and tasks properly coordinated is quite challenging for someone without any experience in this type of leadership role. If anything, it has helped some members of the team notice their own flaws and shortcomings, especially with respect to due dates and organizational skills.

In order to ensure success, the steps to install Caffe should have been taken much sooner than they actually were. This would have given the team more time to post questions to IBM's forum, even if the answers were likely to be of little use. The project is in a relatively stable state at the moment though, and integrating TrueNorth into the existing system is most definitely possible. As it stands, a laptop is being used to emulate TrueNorth in a message passing system with the Raspberry Pi. The Pi controls all of the motor movements and image capturing, and "TrueNorth" accepts image frames and returns tile classifications to the Pi.

Statement of Team Members' Contributions:

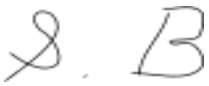
Hollis Bui was a computer science and mathematics major who was originally assigned to work with others on the neural network as well as provide supporting programming assistance, particularly with mathematical calculations. However, due to the challenges faced by the team in terms of scheduling tasks and managing delays, Hollis instead provided a more supporting role overall to the group than working with the project directly. Overall, he completed a program calculating angles for the launcher's aiming and worked mostly on documentation and paperwork as an editor and writer.

Kevin Dunn was a computer science major who was also originally interested to learn the capabilities of current neural network systems. As the project's goals to solve the problems faced by the team shifted, so too did his role in the group: It is fair to say that now, the majority of the work done on GPIO and motor control interfaces have been programmed by Kevin, with assistance from Nick who handled all the wiring and helped in signal control and Shawn who helped with calibration. Because of the hands-on, rigorous testing-based nature of this approach, Kevin worked extensively with both hardware and software to produce visible results for our project and launching darts at the target. Kevin, Shawn and Luke helped create and identify the training dataset as well as create the networking procedure between the Pi and Shawn's laptop in lieu of TrueNorth. Luke also wrote the OpenCV program to identify the center of the target in an image. Nick and Jalen worked to ensure our circuitry worked reliably and would not fail when put to the test.

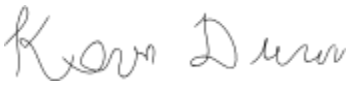
Garrett Massman was the team lead and idealist. At the beginning of the project, the divisions of labor seemed clear, but as we've already mentioned, each person's role in the project was forced to shift as the semester went on. Luke Bechtel provided a secondary leadership role by assisting with 3D printing components, writing the initial Python code upon which all other programs were built, and maintaining a positive attitude through all the project's highs and lows.

Signatures:

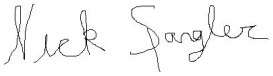
Luke Bechtel

Shawn Bradford 

Hollis Bui  4/19/16

Kevin Dunn  4/24/16

Garrett Massman  4/19/16

Nick Spangler  4/19/16

Jalen Tarvin

Customer Signature:

Customer Signature

4/25/16

Date