CMPE-118 An Introduction to Mechatronics
Final Lab Report

Matthew Luxton
Ariel Anders
Kyle Huey

MAK Attack
University of California, Santa Cruz

March 19, 2012
Winter 2012

Contents

1 Introduction

The final project requirements were to build a 11 x 11 x 11 autonomous robot capable of shooting an enemy robot, maneuvering an 8 x 8 course to the enemys island, and return to the home island without crossing land.

1.1 Team Roles

Matthew Luxton Primary Mechanical and Electrical Engineer, Algorithm

Ariel Anders Systems Engineer, Programming Lead, and Circuit Debugger

Kyle Huey Secondary Mechanical, Electrical, and Programming Engineer

2 System Level Design

1. Prepare for Murphys Law
2. Develop simple modular components for a versatile robotic system
3. Merge integration with development

2.1 Prepare for Murphy's Law

Our group planned for Murphys Law to effect the design of our robot. We knew that creating an intricate, advanced system would be detrimental to implement if any single part of our robot broke.

2.2 Develop simple modular components for a versatile robotic system

Since we expected everything to break and fail, our goal of creating modular components came as a natural result from our first objective. We approached system design with the same methodology as programming abstraction. By developing a uniform chassis to hold the base of our robot together, we could different types of wheels, motors, and shooting mechanisms. Our mechanical design was to insure that replacing parts would be easy in the event that any component broke.
Additionally, we were not afraid to develop simplistic components. Although, this course is heavily influenced by bragging rights, we kept in mind that a functioning robot from simple parts is better than a non-functioning fancy one.

2.3 Merge integration with development

Most people were surprised to see our robot check off as quickly as it did considering the week prior to it, we were not even driving the bot. Although we were joyous to check off, we were not surprised. In fact, we expected check off a little bit sooner... Our trick was not planning for extra integration time at the end of our building and development. This sounds counter-intuitive except for the fact that we planned 5 weeks of integration time. Every time we created a mechanical or electrical component we created test-harnesses in parallel to interface with the micro-controller to test it, and interface it with our end design.

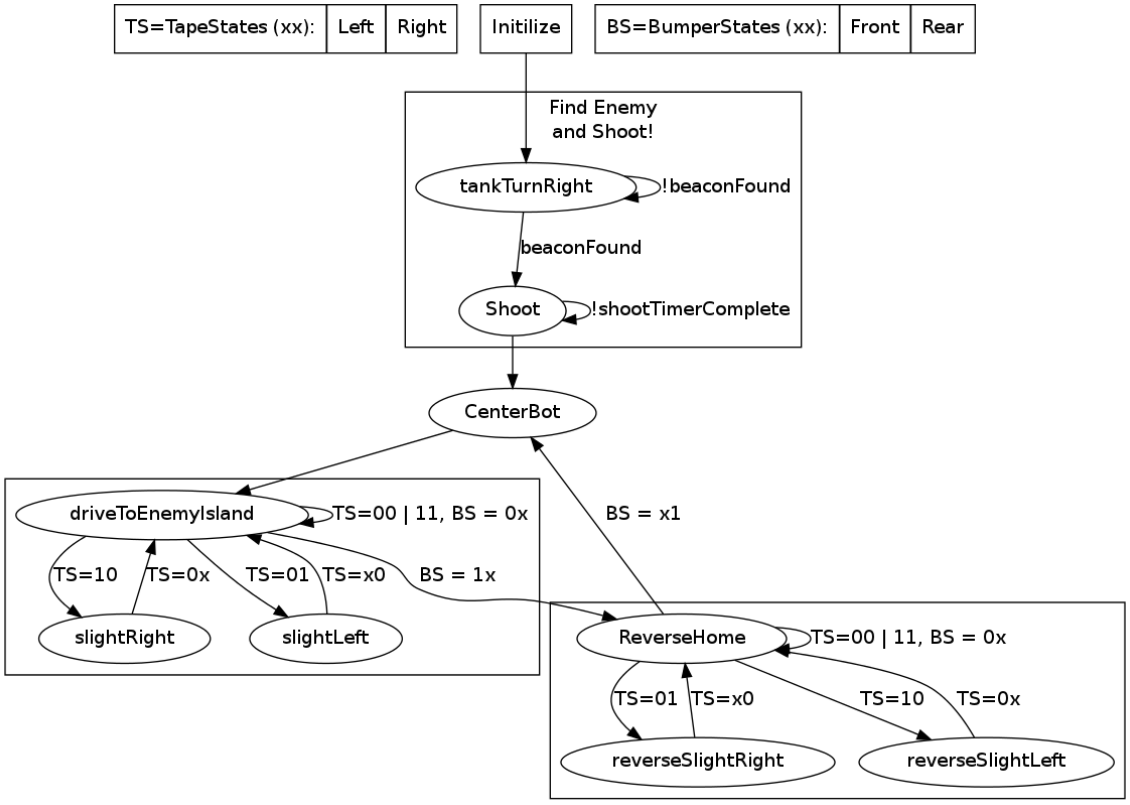

Figure 1: Main State Machine

3 Programming

3.1 State Machine

We had three main approaches for how we would accomplish the task for this project:

1. Line following until we reach the opponent and then line follow home.
2. Zig-zag down the course using the tape boundary and any obstacles as walls.
3. Center the robot, drive to the enemy island, then reverse straight home.

We were planning to implement a mixture of 1 and 2, but the check off date loomed so we decided to go with plan 3 because of its simplicity and the fact that our tape sensors were constantly malfunctioning. Matt and Kyle continued working on fixing the tape sensors and Ari worked on implementing part 3.
The following page shows the state machine for part 3.

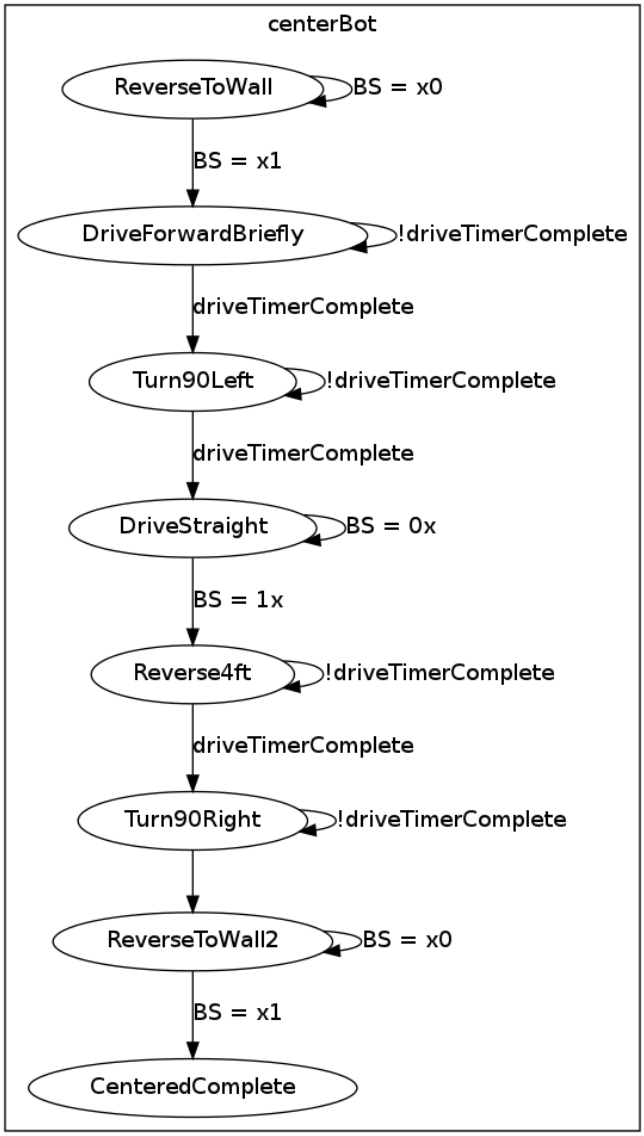

Figure 2: center bot state machine

3.2 Event Detector

The following events were stored using a union-struct data type to set the event flags.

Tape Sensors far left and far right (see sampling section below)

Beacon Filter check if AD input is greater than the BEACONTHRESHOLD

Drive Timer Check if the drive timer function is complete

Bumpers Check if front and back bumpers are triggered

3.2.1 Tape Sensor Sampling State Machine

To get an accurate tape sensor reading, that was not as sensitive to ambient light, we sampled the tape sensor on and off and took reading during the on and off states. The difference between the on and off states determined the tape sensor state. The following diagram shows the state machine for sampling the tape sensors.

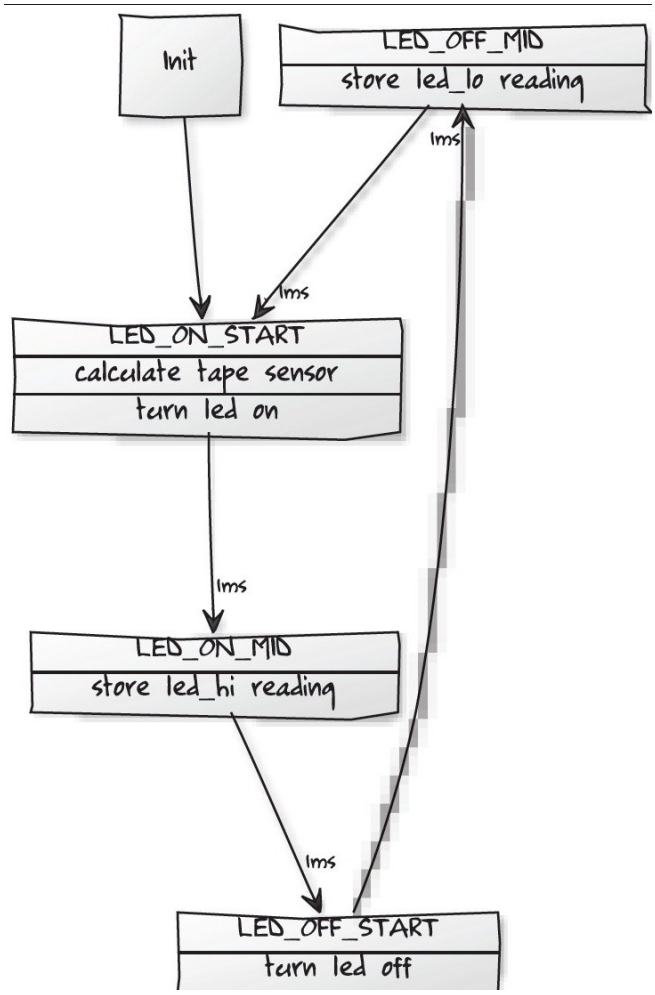

Figure 3: Tape Sensor State Machine

3.3 Functions

3.4 Motor driving states

Tank turn Left Motor: Half Speed, Right Motor: -1 * Half Speed

Forward Left Motor: Full Speed, Right Motor: Full Speed

Reverse Left Motor: -1* Full Speed, Right Motor: -1 *Full Speed

Turn Left Left Motor: Half Speed, Right Motor: Full Speed

Turn Right Left Motor: Full Speed, Right Motor: Half Speed

Beacon Found Left Motor: 0, Right Motor: 0

3.4.1 Timed Functions

The following timed functions used the driving functions for a specific amount of time. We found the time using empirical testing.

1. turn 90 left
2. turn 90 right
3. reverse 4 feet
4. drive forward briefly, approx. 3 inches
5. shoot for 4.5 seconds

These driving functions used the same timer using the timers.h package. By using the same timer, we were able to make a single timer complete flag in our event detector. This greatly reduced the complexity of programming our robot.

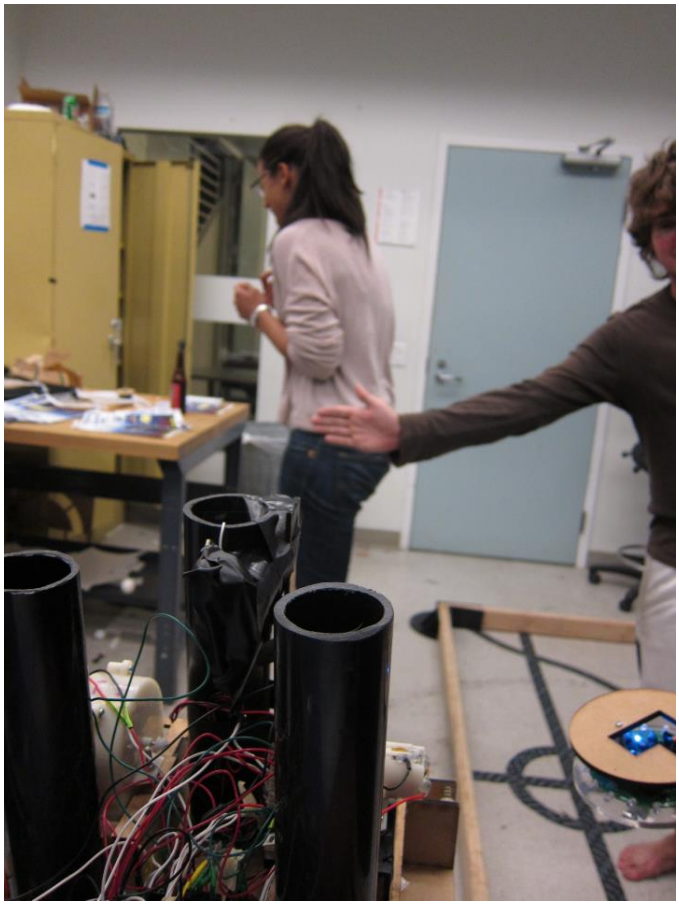

Figure 4: Ari hit from an over-powered ping pong

3.5 Battery Voltage

We initially tested our robot using a very long power cord to the ATX power source. Once we had completed our algorithm, we placed in the batteries and re-tested our robot. It did not work because a single 7.2 volt battery did not have enough potential to drive the motors from the H-Bridge (under 8.2v). Then, we placed two batteries in series for a total of 14.4 volts of potential. Our bot drove, but drove much too fast and shot ping pongs extremely hard. (See image)

To fix this we added some code to the intilization state and to the driving functions. We adjusted the PWM signals to drive the motors based off the porportion of battery voltage. Our desired battery voltage was 12 volts. Since we have 10 bits of resolution and an analog reference voltage of 33volts, the desired voltage with respect to the micocontroller is
$(2^{10} - 1) * \frac{12v}{33v} = 372$
Then, if the battery voltage is $V_{batInput}$. We can find the porportion of battery voltage to the desired 12 volt input:
$V_{batInput} * p = 372 \rightarrow p = \frac{372}{V_{batInput}}$ The rest was easy, since PWM works off porportion, we included this proportial constant when driving the motors:

Drive($leftMotor * 372/V_{batInput}, rightMotor * 372/V_{batInput}$);

4 Mechanical Design

4.1 Modular Chassis

We began the process of building our robot with discussion. We got together with copies of our midterms and began to formulate ideas for what would eventually become our final project. We decided that a good starting point would be a basic modular chassis, to which we could attach a wide variety of other components. The chassis itself would house two motors, the H-Bridge in between said motors, and the batteries. It would be structurally sound on its own and capable of driving, but set up to accept a top plate of whatever was required. We came up with the following design:

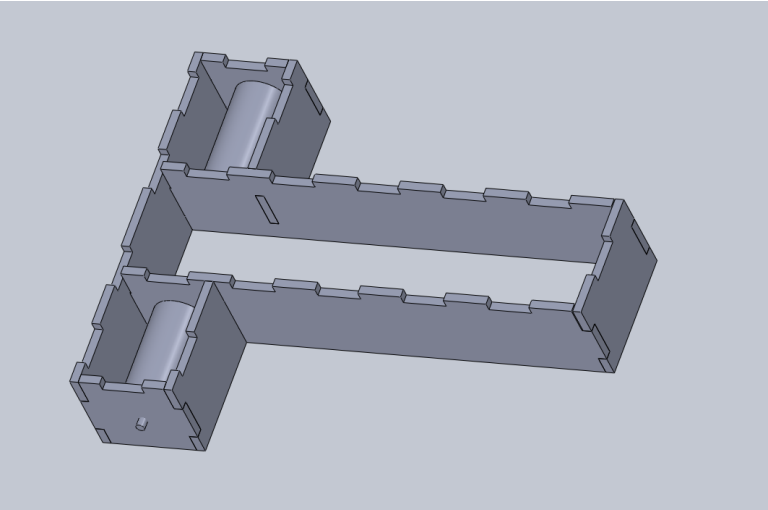

Figure 5: Basic Modular Chassis Assembly

Since we still did not have exact dimensions on the motors, we substituted some cylinders to approximate them for the time being. At this point we began to come up with ideas for ball launchers. Initially, we planned on having three sets of launchers, two forward facing and two banks of side by side one shot side facing launchers. The side facing launcher idea was eventually scrapped, as it required three beacon sensors and a number of added components, and time was becoming a more and more precious commodity. Below is a drawing with the launchers present.

4.2 Shooting Mechanisms

The side launchers eventually became battery space, and the original battery/H-bridge space became a place to conceal wiring. In addition, the swept 90 degree ball hoppers were replaced with straight pipe, as they were much easier to fit on the bot. A tower in the original model was added to house the beacon sensor, but due to a printing error we decided to instead house the beacon in a piece of ABS pipe similar to the ones used to hold ping pong balls. As we assembled sensors, we realized it was going to be necessary to house them, and therefor we should incorporate space for them in our mechanical design. We added a wide front bump plate attached with light springs to trigger the switch that was our bump sensor. This plate extended past the wheels and anticipated though ill fated side shooters to make sure that no part of our robot stuck out further than we could detect a bump. A plate was also added to

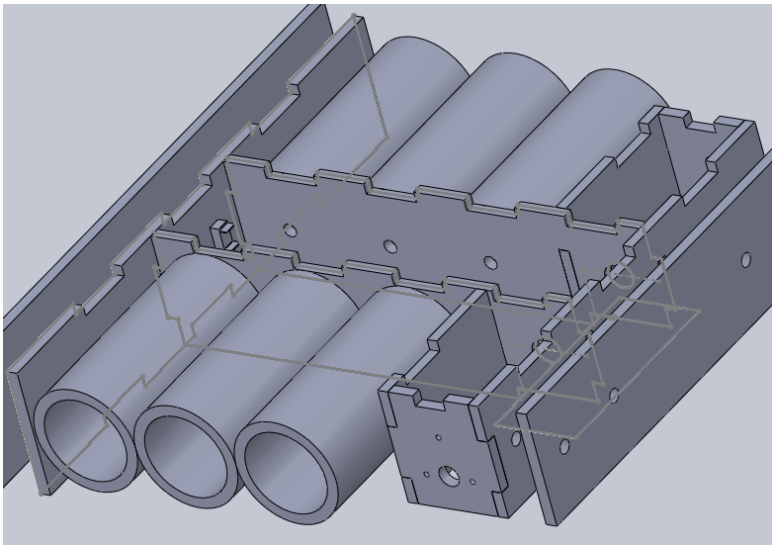

Figure 6: Design with Side Launchers

the bottom of the robot, to facilitate the mounting of tape sensors. Space was set aside for a bank of 2 or three sensors in the middle and two outer sensors that would be mounted farther out towards the edges. We decided to include all necessary circuitry/power distribution locally on the sensors rather that centralising, as we felt that a more modular design would be easier to troubleshoot and potentially replace if we ran into problems, which proved accurate. A rear bumper was also added in a similar manner to the front bumper, though the rear bumper did not extend out as far past the wheels as we anticipated collisions from the back would be predominantly from intentional bumps to establish location rather than from incidental bumps caused by unanticipated obstacles.

Our ping-pong ball launchers were relatively simple in construction. The toy motors we bought as the launching mechanism had semi circular channels in them already, which would make sure the balls we launched traveled straight. We constructed a simple mount for the motor and left the rest to the geometry of the launcher, making sure that the ball and the motor had between and 1/16 overlap so the motor channel directed the ball properly.

These launchers were initially fed by by a set of micro servos that would fit into a cut off block and sit on top of the base plate. The servos would push a ball into the launcher while holding the next ball up in the hopper, then retract, letting the ball it was holding up drop while pushing in the next ball and holding yet another ball and reciprocate in this fashion until all the balls were fired. Sadly, the micro servos died in a horrible accident, and we were forced to mount (rather poorly) some much larger servos to do the same thing. This was not the most attractive feature of our robot, but was necessary and seemed to work okay. We mounted the Pic between the hoppers and behind the beacon sensor. This seemed logical, as it would protect it from a front side barrage while allowing us easy access to it. Also, this gave us a good amount of separation between it and anything pulling a lot of current (i.e. motors and H-bridge) while keeping it fairly central to the design itself. Additional holes were added for wire routing, and the pipe holding the beacon became a sort of unofficial conduit with wires from both above and beneath routing through it.

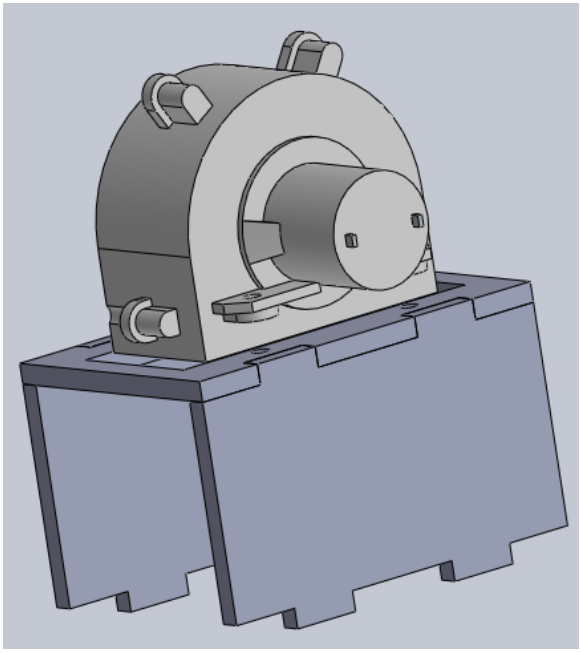

Figure 7: Ping Pong Ball Launchers

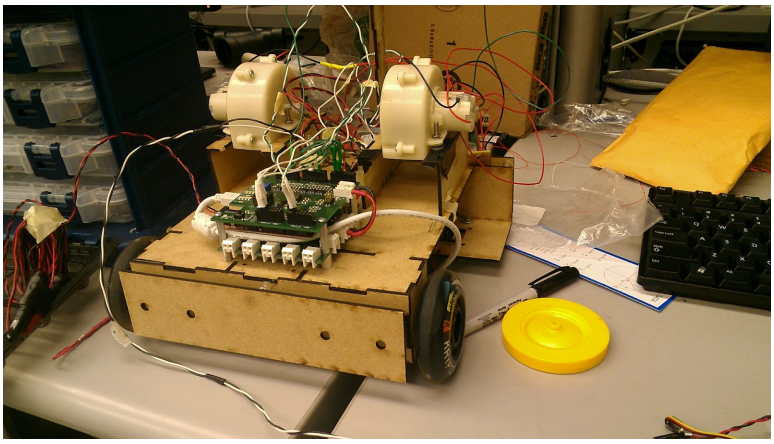

Figure 8: MAK ATTACK in construction

5 Electrical Design

5.1 Beacon Filter Circuit

We began with the beacon sensor. There is still some disagreement as to the functionality of this component (Kyle says it works great Matt says it lacks a low pass stage) but it was time for the circuit so we soldered it up and threw it on the bot. We set some threshold values for the beacon and tape sensors that would be adjusted more later, but for the purpose of testing we were able to sense tape and beacons. The bump sensors were basic on off devices and required no thresholds.

5.2 Tape Sensors

This circuit was relatively simple, as the two main components were packaged together. The only tricky part was making sure the current supplied to the parts of the circuit was ample to overcome the voltage drop in the diode and the bias current of the photo transistor, or so we thought. It turns out, it is possible to burn components out with a soldering iron that is too hot. The LEDs on the first set of sensors were fried, and not emitting light. When the identical circuit on the breadboard worked, we used the built in diode checker on the DMM and found that our LEDs were at fault. The tape sensors were quickly replaced and we were able to move forward.

5.3 Bump Sensors

The bump sensors were simple. We ran 12V to a switch in line with a 10M resistor. We took a reading directly below the open switch. This value would read low unless the switch closed, in which case it would read high. The large resistor ensured that we did not sink to much current through this circuit during the switched stage.

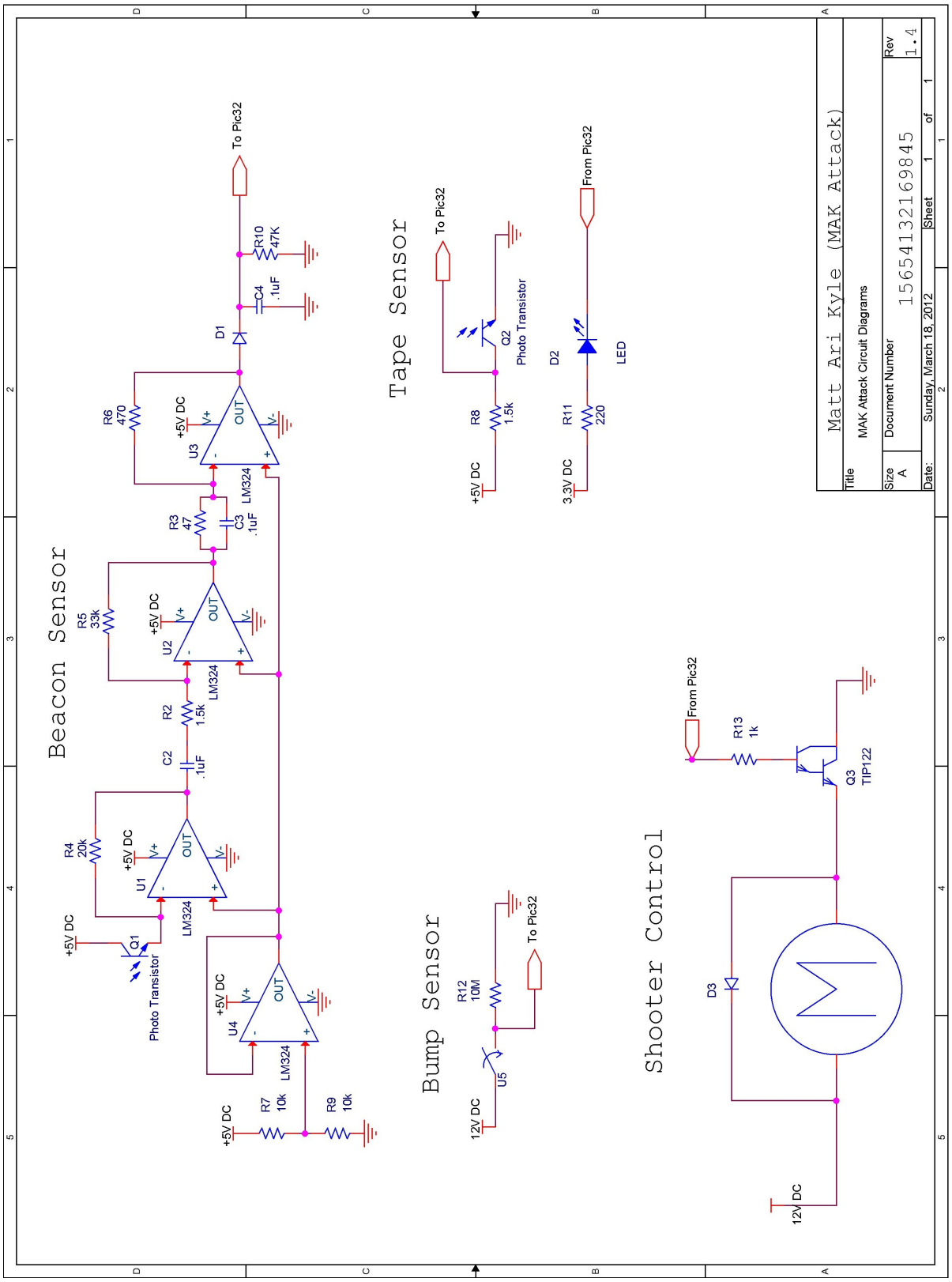

Figure 9: Design with Side Launchers

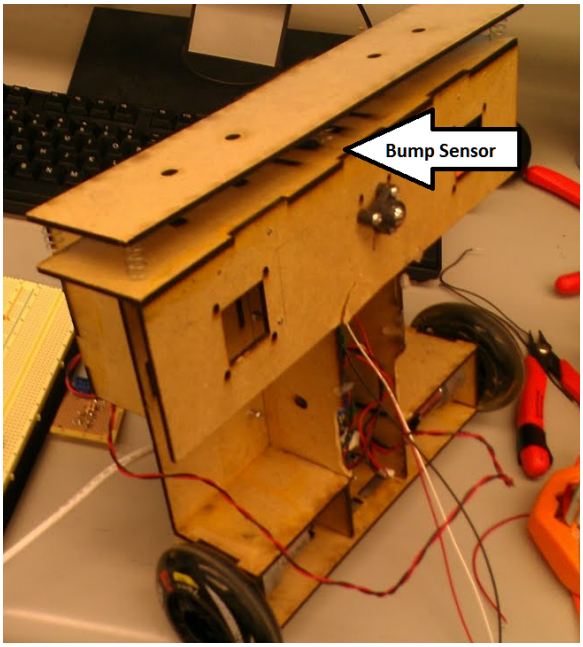

Figure 10: Bump sensor circuit integrated with mechanical design

5.4 Motors

The last circuit we had to come up with was a control for the toy motors. This circuit was based around the TIP 122 IC and took in a PWM signal from the PIC 32 which it used to forward bias a larger current to the launcher motors. This circuit also included a diode to help prevent back EMF into our system which could create problems in the form of noise.

5.5 Driving Motors

We drove the dc driving motors using using the H-Bridge.

6 Integration

- We began by mounting the motors to our drive chassis, and testing the chassis ability to drive.
- After, we mounted the front and back bumpers and ensured they functioned. A problem, which was easy to fix was we forgot to account for the size of the back wheels, which prevented the back bumper from triggering. Rather than redesign the robot, we elected to add additional material to the back which fixed the problem and saved our group time for testing other components.
- With the robot driving and detecting bumps, we mounted the tape sensors, and through tests determined the thresholds necessary for when the robot was on or off black tape. Our initial plan was to mount the sensors in the back of the robot, but after deciding to change our strategy for driving, moved the tape sensors farther up towards the middle of the robot. Similarly to the situation with the back bumper, rather than cut a new robot, we decided to manually alter our robot to fit the tape sensors in the middle. An

issue with the tape sensors was the threshold values needed to be constantly re-calibrated depending on the lighting conditions.

- After modular testing of the tape sensors, we mounted the IR Sensor and ran tests to determine the threshold for when the robot spots the IR beacon. These tests initially were fairly quick to do, however, like the tape sensors, the threshold value had to be re-calibrated depending on the lighting conditions.

- The final step was to mount and test the ping pong ball shooters. Initially, the noise generated by the toy motors interfered with our signal to the ball feeding servos, however, after wrapping a ground wire around the signal wires, the ball shooting system worked well. The hardest part of testing the shooters was having to replace the servos when we accidentally broke them.

- After performing modular tests on each of the components and prior to integrating the components together, we wrapped power and ground wires together and made the wiring neater, to limit the potential for noise and to make debugging easier.

- The integration of all the components did not take long as we expected, which was because we spent more time planning and accounting for what could go wrong when we integrated the circuits together. Most problems we encountered were the need to recalibrate the tape sensors and the IR sensor thresholds, and the need to change the timer values for our drive times, which were all software related.

7 Results and Challenges

7.1 What Worked

The front and back bump sensors were well designed and implemented, as their sensitivity was exactly how we planned. They were never accidentally triggered and were fairly sensitive to minor bumps.

The use of software to determine whether our robot detected the IR beacon allowed us to detect the beacon with great precision because we could change the thresholds to account for external lighting conditions. As a result, we were able to produce a accurate ping pong ball shooter mechanism which hit the stationary robot consistently with more than two ping pong balls.

7.2 What Did Not Work

Most equipment in lab.

Although the use of software for setting the thresholds worked well for detecting the IR beacon, every time we used the filter it needed to be re-calibrated. Our bot did not perform well at the competition because we did not have the opportunity to recalibrate the filter.

The tape sensors implemented did not work great, but were passable if calibrated prior to use.

7.3 Challenges

- Kyle had a tendency to accidentally solder power and ground together because of use of damaged equipment, lack of experience soldering, and laziness. (i.e. when building

the Shooter Control circuit (TIP122)). Fortunately for Kyle, Ari has experience making shoddily designed circuits work. The majority of the circuits we were able to debug, troubleshoot, and figure out where the shorts were and fix them. This was a good learning experience for each member of our group because Ari got practice debugging hardware, Kyle got practice building circuits, and Matt had time to take a nap.

- Though we were able to get the tape sensors working, we had trouble getting them to function. Our first set of tape sensors had blown LEDs, and it was difficult to determine the problem as we cannot see in the IR spectrum. However, the second set worked fairly well, and using software to determine the thresholds allowed our robot to detect black tape.

- The shooting mechanism consisted of a servo feeding a ball at a time to the toy motor. If there were too many balls fed to the toy motor, the system would jam. This was unfortunate because we broke a servo during one of these jams. The servo kept trying to push a ball into the toy motor and eventually ruined its gear set. Then, we found a cool connector that appeared to connect two servo motor outputs with one input. This was not the case, in fact, it blew up our last two working servos (we bought three for just in case). We ordered two more micro-pro servos, but did not get them in time to use on our robot. We ended up using two of the larger servos from Professor Elkaim instead. The larger parts were more robust and worked well as a feeding mechanism, unfortunately, we did not account for using the larger servos in our original mechanical design and resorted to gluing the servos onto our bot.

- On the weekend prior to check-off Ari visited University of Michigan: she departed early Thursday morning and returned late Saturday night. We carefully planned our development to account for her absence. As the system engineer and programming specialist in our group, Ari was least concerned about the integration, but rather the development of working, physical mechanical parts. She had a quadruple- all-nighter prior to her trip, which she spent creating numerous test harnesses for all physical parts interfacing with the micro-controller and spent the rest of her time debugging hardware. Ironically, Aris absence could have been one of the best challenges for our group. We were forced to communicate, plan, and implement our work schedule perfectly in order to accomplish our objective. In reflection, the previous group projects weve worked on and gained communication skills and improved project management are minute in comparison; in utter honestly we can say we had the most communication and furthest developed project development plan in our engineering endeavors to date and probably between our classmates as well.

8 Conclusion

Itemizing the things that worked and did not work may seem like a clear indicator for our projects success; however, the challenges we encountered in the course and our path of overcoming or finding a circuitous path around them are what makes the biggest impact in our minds eye. We feel the final lab has successfully taught us how to go about completing a project in an organized manner by following the design process of planning, designing, building, testing, integrating and debugging. Though we spent many hours in lab, we enjoyed spending our time in lab and our efforts culminating in an autonomous robot meeting the required specifi-

cations. After completing this lab, we have gained experience following the design process and are comfortable using this process again.