EEL 5666

Intelligent Machines Design Laboratory

Spring 2005

Final Report

μCHIP

(Micro-Controlled High-tech Independent Putter)

Instructor: Dr. Arroyo

Student: Rob Hamersma

Date: 4/19/2005

Table of Contents:

Abstract ……. 3

Introduction ……... 4

Integrated System …….. 4

Mobile Platform ……. 7

Actuation …… 7

Sensors ……...

Behaviors ……...

Experimental Layout ……..

Conclusion ……..

Documentation ……

Appendices …….…….

Abstract

μCHIP (Micro-Controlled Hi-tech Independent Putter, or just CHIP for short) is an autonomous miniature-golf-playing robot that can consistently defeat most humans at putting. Of course, he needs to use a red ball, and be on his own green, indoors, with adequate lighting, and within 6 feet of the hole, but that's not too much to ask, is it? How does he do it? CHIP uses a small digital camera to search for and recognize his golf ball, which he then approaches and straddles between his two wheels. Using the camera again, he spins in place over the ball to find the hole, which is marked by a larger red ball mounted over it. Calculating the distance to the hole based on the perceived size of the hole marker ball, he then swings his putter and putts the ball hard enough to get a hole in one every time! Ok, not every time, but with a consistency that remains unmatched by your average would-be putt-putter. CHIP is completely autonomous. He will keep searching for balls and putting them into the hole until he runs out of balls, or you turn him off, or his battery runs out. He will even pull a ball away from a wall if it is too close for him to putt and then putt it in. And while he's not putting he spends his time wandering around, avoiding obstacles and exploring the room. He's the perfect techno-pet!

Introduction

The goal of this project was to design and develop a robot that plays miniature golf (or a sufficiently simplified version of miniature golf) autonomously. I chose this objective because it was complicated enough to be challenging, yet simple and scalable enough to provide a reasonable chance of success given my budget and schedule. Also, I found few other robots published online that had attempted this, and none that appeared to operate consistently on the scale that I hoped to achieve. My ultimate aim is to create a robot that will be able to compete on a regular outdoor miniature golf course, but for now, the robot is able to perform the following four different functions: (1) It can locate and approach a golf ball that has been randomly placed near it, (2) determine the direction of the hole and turn to face that direction, (3) compute the distance to the hole, and (4) strike the ball with the necessary force to sink the putt. The robot I developed to accomplish this task I've called μCHIP (Micro-Controlled High-tech Independent Putter), or simply CHIP, a common nickname that is also a term used in both in golf and electronics.

Integrated System

CHIP is essentially an independent electromechanical system comprised of a brain (microcontroller), eyes (digital camera), ears (sonar module), antennae (bump switches), actuators (motors for the wheels and putter, and a servo for the head), a user interface (buttons and an LCD screen), and a power source (rechargeable battery) all mounted on a platform. (CHIP is about 12" long, 12" wide, and 9" tall.) His brain is an Atmel ATmega128 microcontroller mounted on a MAVRIC-IIB development board, which is basically a miniature computer that processes the data it receives from the various sensors

and buttons, makes decisions based on that data, and issues commands to the actuators and displays information on the LCD screen (Fig. 1).

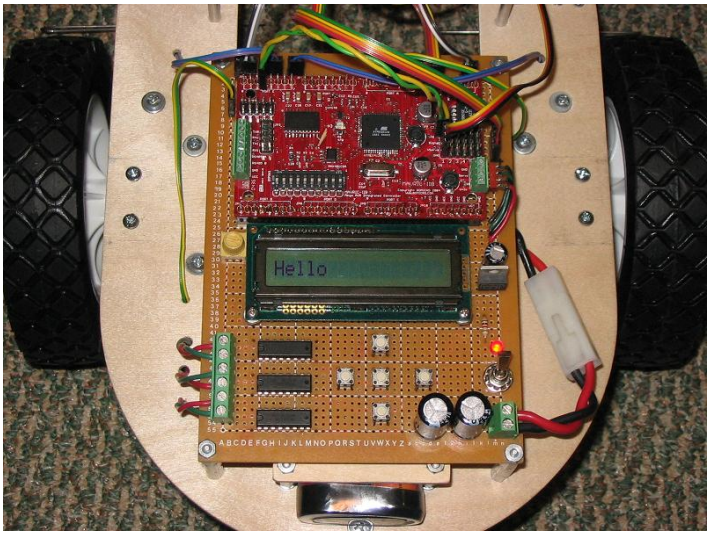


Figure 1 - Circuit board with LCD and buttons

CHIP uses a CMUcam, a small color digital camera module, to locate and approach the ball, identify the direction of the hole, and determine the distance he needs to putt (Fig. 2).

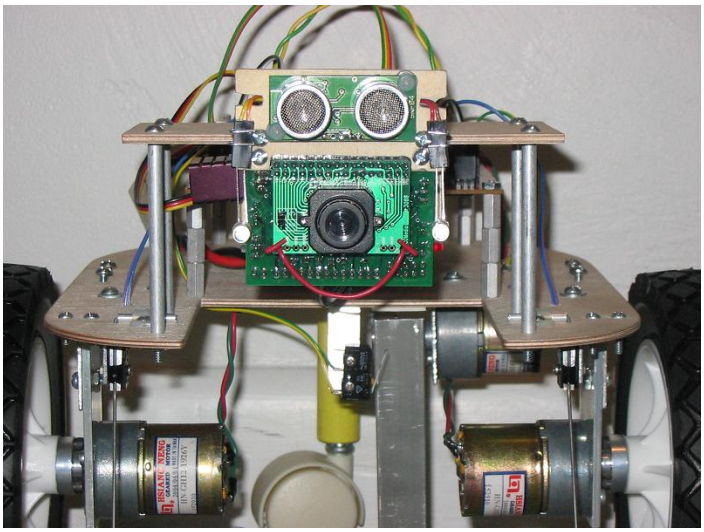


Figure 2 - CMUcam (nose) and Sonar (eyes)

He can also detect objects by using an SRF04 Ultrasonic Range Finder to sense objects at a distance (Fig. 2), and two bump switches with extended wire actuators to sense objects up close (Fig. 2,3).

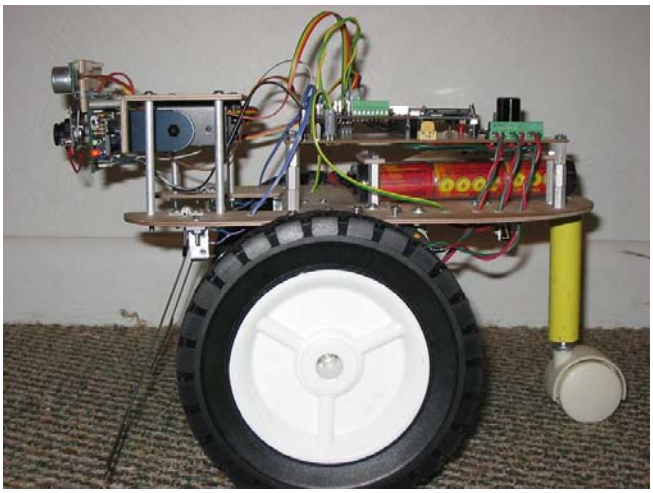


Figure 3 - Showing battery, servo, bump switches

To interact physically with his environment, he uses two DC gear motors to power his two drive wheels, another to swing his putter (Fig. 4), and a servo motor to tilt his head (camera and sonar modules) up and down. The entire system is powered by a 3000 mAh, 7.2V NiMH rechargeable battery pack (Fig. 3).

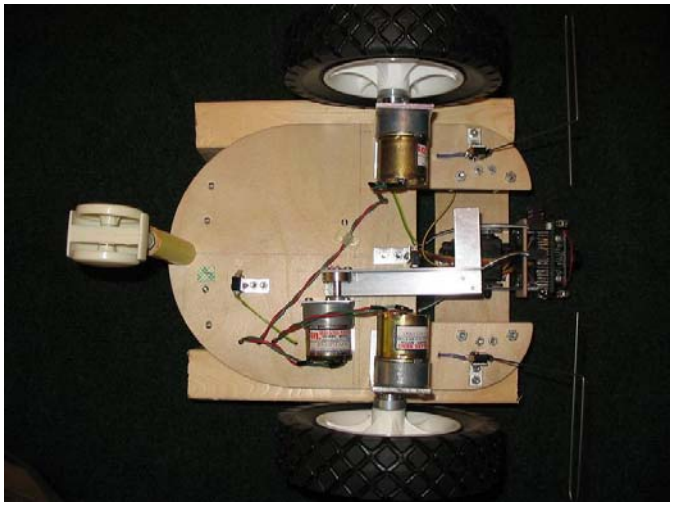


Figure 4 - Underside

Mobile Platform

The chassis consists of one main platform on which all the components are mounted. I decided to use a 2-wheel differential drive design with a rear caster for support so CHIP would be able to spin in place and maneuver easily. The main design consideration for the structure of the chassis was that CHIP needs to be able to position the ball directly between the two drive wheels. This enables him to find and approach the ball directly and then spin in place around the ball (driving the motors in opposite directions) to locate and face the hole. This is a much simpler way to line up a shot than figuring out how to drive around the ball to approach it from the correct angle. To accomplish this, the platform is raised about 5-1/2 inches off the ground to accommodate the 4-inch putting arm (Fig. 4), and is cut away in the front in the middle so that the camera and servo can be mounted above the platform and have room to tilt down and follow the ball underneath the robot (Fig 2,3). Since the platform is so high, the drive motors and wheels needed to be mounted on brackets that lowered them down from the platform about 2-1/2 inches. The overall size of the platform was designed so that the 4-inch putter arm has room to swing freely back and forth underneath and also that the caster wheel and drive wheels follow close to the same arc so CHIP won't hit anything while spinning.

Actuation

The motors used to power the wheels and the putter arm are DC gear motors that were chosen primarily for their small size, since I needed to have room between the wheels to swing the putter arm. I'm using one SN754410 H-bridge chip to drive each motor because they can operate at low voltages (down to 4.5V), handle up to 1A output current,

which is greater than the stall current of the drive motors, and provide for motor braking. The largest drive wheels I could easily find to use were 6-inch lawnmower wheels, and these had to be mounted on brackets to lower them down from the platform (Fig. 2,4). At first I had thought I would like CHIP to have relatively large wheels so he could scoot around quickly if necessary, but it turned out that I wished I had used smaller wheels because they would be easier to control precisely and require less torque from the motors. As it is, these wheels worked well enough, but I ended up having to use slower motors with more torque than the ones I had originally. The putter motor was chosen to be much faster than the drive motors so that it could hit a golf ball at least 5 feet (and size of lab space at home, and reasonable distance for demonstrations). I chose a 175 rpm motor and then experimentally determined the length of the putter arm to be 4 inches in order to putt 6 feet.

Sensors

CHIP uses a variety of sensors to collect data from his environment in order to make the right decisions and accomplish his goal:

Bump Sensors:

At the front of the platform, there are two mechanical lever switches with extended wire actuators positioned so that they will be activated first if CHIP gets too close to an object (Fig 2,3). When one of the switches is activated, the microcontroller immediately sends the appropriate signals to the drive motors to cause CHIP to stop, back away from the object, and turn away from it before continuing his task. At first I tried gluing the wires to

the switch levers, but they would pop off when they were twisted, so I had to mount the wires independently from the switches, and that seems to have worked well.

Sonar:

Another sensor that CHIP uses to avoid obstacles is his sonar module, the SRF04 Ultrasonic Range Finder.

Figure 5 - SRF04 Ultrasonic Range Finder

This unit uses a simple interface to send an ultrasonic pulse and detect the first echo it receives. The microcontroller just needs to trigger the sonic burst with a short pulse and then measure the time that the echo line is held high waiting for an echo and compute the distance to the nearest object based on the speed of sound through air (73.746 µsec/in.).

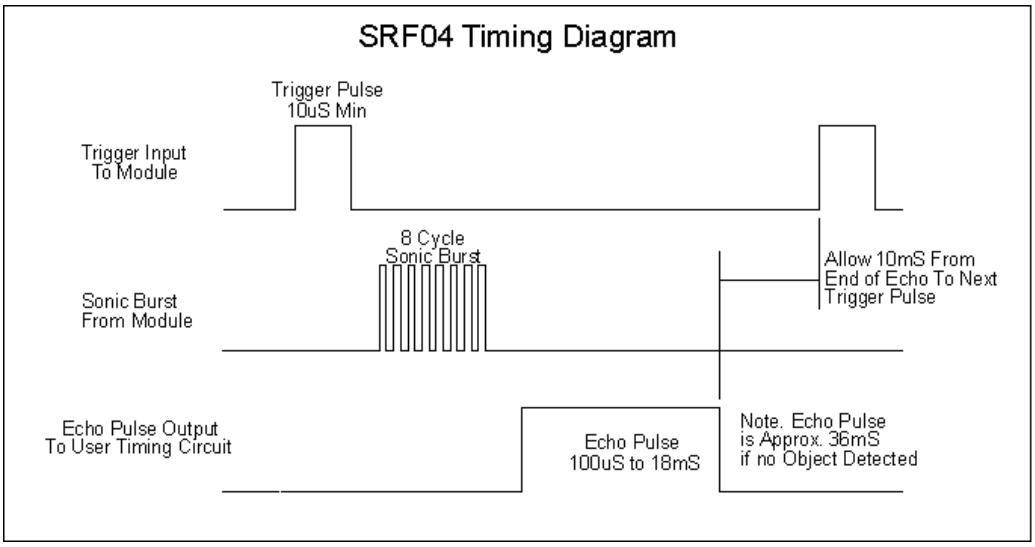


Figure 6 - SRF04 Timing Diagram

This sensor is used to detect objects from a distance and minimize reliance on the bump switches for obstacle avoidance.

CMUcam:

The most complex sensor that CHIP will use is the CMOS camera module, the CMUcam (named for Carnegie Mellon University where it was developed). It consists of a small, relatively low-cost OV6620 Omnivision CMOS digital camera coupled with an SX28 microcontroller to provide simple high-level image processing data for end-user applications. These data can be accessed via RS-232 serial communication protocol by another microcontroller or a PC. The CMUcam can identify and track colored objects that fall within given RGB (Red, Green, Blue) values at up to 17 frames per second with 80 x 143 resolution. It can also provide information about the object such as the center, size (in pixels), and the confidence of detection. The camera is mounted on a servo so that it can be aimed at any angle from 0º (looking straight ahead) to 120º (looking back under the platform between the drive wheels).

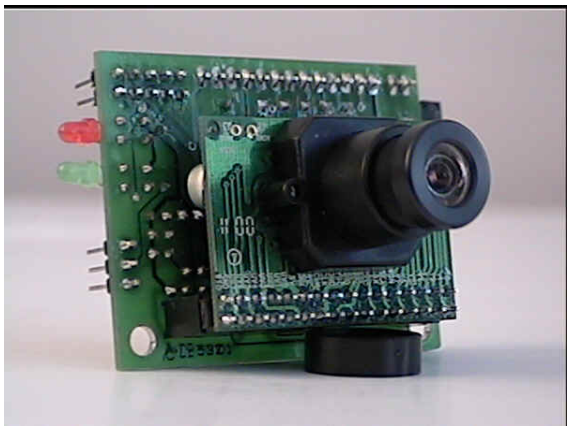


Figure 7 – CMUcam

Interfacing the CMUcam with the microcontroller was fairly straightforward and I was able to take advantage of most of the capabilities it had to offer. Communicating with the CMUcam is easily accomplished by using the built-in UART in the microcontroller and connecting directly to the camera's TTL level serial port. Setting the baud rate to the fastest rate of 115,200 works well and maximizes the data throughput. The configuration settings CHIP uses are: raw mode, middle mass mode, YCrCb color mode for tracking the golf ball, and RGB color mode for tracking the hole marker ball. Raw mode is used to communicate with the CMUcam in actual data values instead of readable ASCII characters, and middle mass mode provides information about the center of the tracked object in the data packet. Unfortunately, the CMUcam only seems to track reddish colors best, at least on varying backgrounds. Most other colors seem to have a fair amount of red, green, and blue, which make them harder to isolate and distinguish. YCrCb mode uses a different color space definition than the more widely known RGB definition, and is less affected by changes in illumination. This makes it easier to track a golf ball as it goes from the open green to the shadow underneath the robot. To help the camera track the ball when it has trouble and when centering over it, I have also mounted two ultrabright white LEDs on either side of the camera to provide lighting when necessary. These turn on during final centering and readjusting for consistency, and also if the camera loses track of the ball while approaching it. Once it is looking for or tracking a color, it sends a constant stream of data packets which include the center of the color object in x and y, the dimensions of a box surrounding the object, the (scaled) number of pixels comprising the object, and the confidence level. I set up the UART to trigger an interrupt whenever a new byte is received and used the packet starting byte (always 255) to parse the data into global

variables. This way, the main program can have access to the latest data at any time without having to request it. Operating in this way at 115,200 baud, CHIP can react fast enough to track objects in real time while moving slowly.

Behaviors

The overall putting process is broken down into a number of behaviors that each need to be performed in order to get the ball into the hole.

First, the drive motors turn the wheels in opposite directions to spin CHIP around in place while the servo positions the head to aim down in front of the platform and the camera looks for a red golf ball based on its color and size. If no ball is found after one revolution, the servo tilts the head up slightly so that the camera can look farther away and another revolution is made. Unfortunately, CHIP can tend to get easily distracted by other reddish objects that fall within his view, most notably any reddish flooring or red or brown objects lying just beyond the green. I have been able to eliminate most of these false positives by filtering based on the maximum expected size of a golf ball when viewed from a given angle, and also based on the confidence level provided by the CMUcam. However, when the ball is near the edge of the green next to a red carpet, the camera lumps them together as one object and CHIP ignores it. If he cannot find a ball in his immediate area he will wander to another location by moving forward, turning randomly, and moving forward again (while avoiding obstacles), and start looking for a ball again.

Once a ball is found, CHIP uses the servo to keep the camera focused on the ball while he approaches it, correcting any misalignment in the x (horizontal) direction by temporarily braking either the left or right wheel. The camera watches the ball until it is

centered underneath the platform so that it rests squarely between the drive wheels. If the ball is lost from view at any point during this process, the camera looks up and down around the last known position and then looks again with the LEDs turned on if it still cannot be detected. This has proven to be a very reliable method to regain a lock on the ball whenever the camera loses track of it.

CHIP then lifts his head and looks straight ahead with the camera while spinning in place around the ball until he recognizes the hole marker, the larger red ball mounted over the hole. Once again distractions are filtered by confidence level and also by the expected height of the marker. It is necessary to remove the previously sunk ball from the hole where it is visible and will confuse CHIP as he is looking for the hole marker, which is a drawback to the filtering process. Once the hole is found and centered, the data from the camera is analyzed to compute the distance to the hole based on the perceived size of the hole marker. The camera then looks back down at the ball and CHIP adjusts himself backward if the ball is too far behind the "sweet spot". (If the ball is too far back, the putter won't hit the ball as far.)

After making any necessary adjustments, he lifts his head again and the putter swings, striking the ball with the calculated speed required to putt the ball the distance to the hole.

Another behavior developed recently is for when the ball is located close to a wall. If either of the bump switches are activated while CHIP is centering over the ball or spinning to look for the hole, he will turn slightly and back up, dragging the ball away from the wall with one of his bump wires. Then, when he finds the ball again, he will have enough room to maneuver around it to putt. This is similar to the rule in miniature golf,

whereby the player may move his or her ball away from a wall or object the distance of one club head length, in order to be able to putt it.

Experimental Layout

Most of the values used to track the colors of the ball and the hole marker and to do the necessary filtering were done by trial-and-error experimentation since the data from the CMUcam tends to be relatively jittery and inconsistent and hard to test directly. It seemed easier to play with the different settings and values and get a feel for what would work and what would not than set up detailed experiments which would need to be relatively complex in order to yield any significant data.

I did, however, measure and test the putting speed necessary at different distances from the hole, as measured by the perceived size of the hole marker. I determined the minimum necessary putting speed at eight different distances, and plotted them out in an Excel spreadsheet. I then did a second-order polynomial regression on the data, since it looked to be a squared-type function and tweaked the coefficients until I had an equation that would work for all distances (Fig. 8). CHIP uses this equation to calculate how hard to hit the ball every time he lines up for a shot. As you can see from the graph, the speed is guaranteed to be 100 (pwm at 100%) anytime the size of the ball is less than or equal to 8, and the speed will never be less than 40, which is suitable for putts right up next to the hole. The values in the middle are a little above what they need to be (measured) but it is better to hit the ball too hard than too softly.

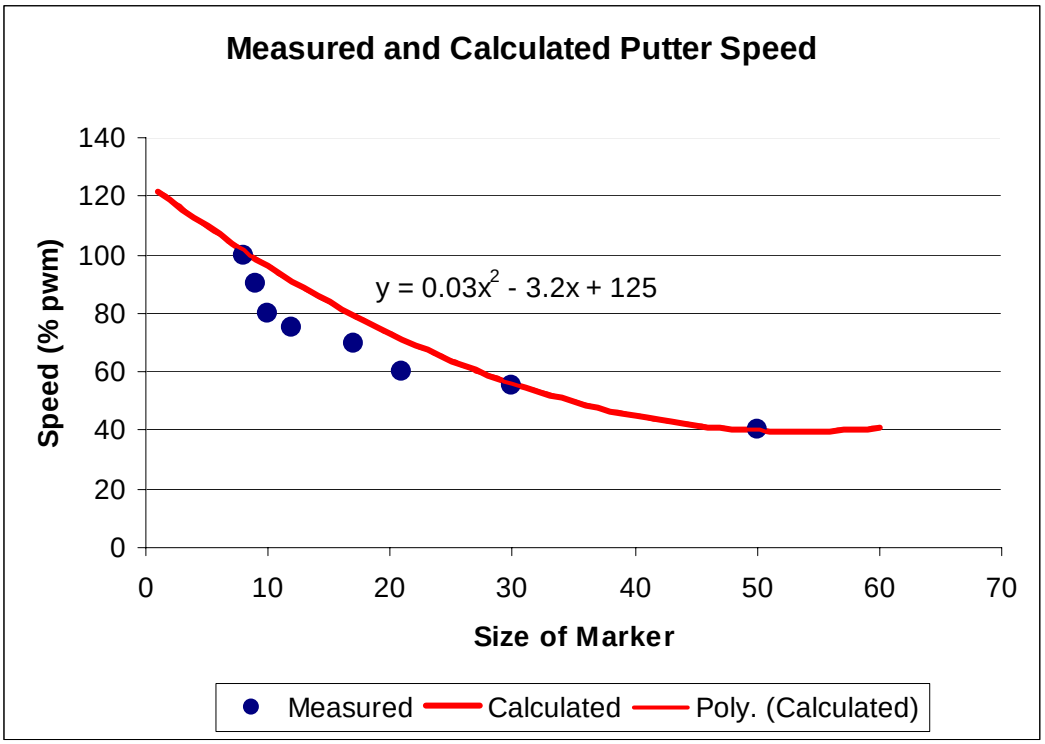


Figure 8 - Putting Speed Experiment

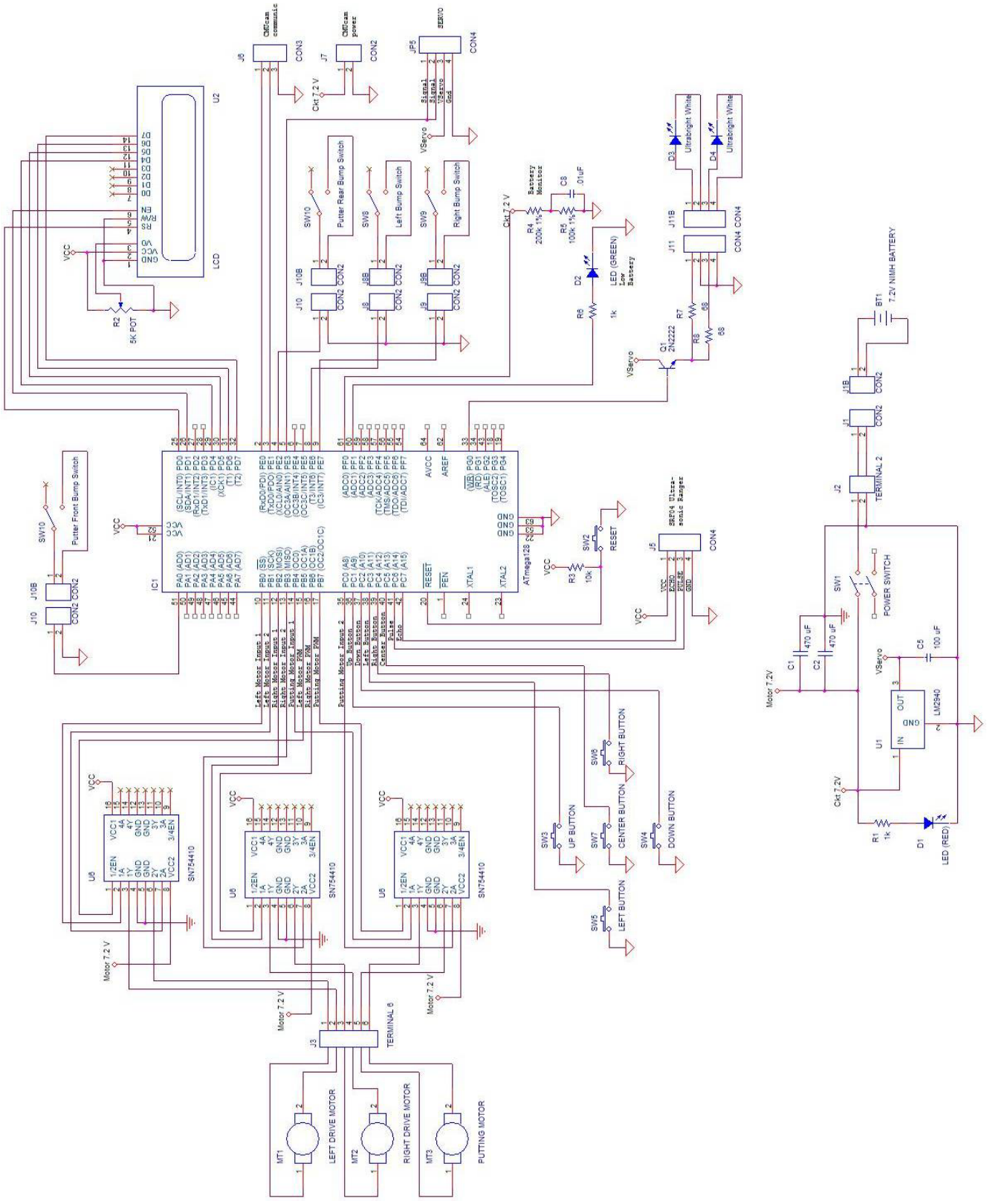
Conclusion

On the whole, I think CHIP accomplishes all the objectives I set out to achieve with him. He can find balls on his own, center over them, find the hole, and putt the ball in with acceptable regularity. He usually will sink about 70% of the putts, and most of the misses are fairly close. He will still get distracted by the right combination of colors in the right places, but he does a pretty good job of trying to ignore them, and given more time to adjust the program parameters, I think he could be even better. Sometimes he can be a bit sloppy centering the ball and looking for the hole, although this would probably be a situation where smaller wheels would allow for tighter tolerances on the centering ranges and greater precision for positioning. I would prefer not to use any slower motors as CHIP's top speed is fairly slow right now, but motors with more torque could potentially

be driven at slower speeds without stalling, which would also improve precision. I will probably eventually replace his 1/8-inch plywood platform with 1/4-inch polycarbonate, which will look better and be less flexible and sturdier for mounting the wheels. Also, the caster wheel tends to add some unwanted lateral motion when making small adjustments perpendicular to the direction it is facing (eg. ball readjustment), so I would like to replace it with a ball-in-socket type or something similar. I'm glad I chose to use the CMUcam, although it can be a bit temperamental. Given adequate lighting it does work quite well, as long as you don't ask too much of it. For better reliability, I would probably rather use an IR and/or sonar beacon at the hole, but it would be hard to match the precision you can get with the CMUcam when it working well. One area that exceeded my expectations was the ball-repositioning behavior for when the ball is near a wall. I didn't expect to be able to resolve that issue quite so easily.

I am pleased with the progress I made and the results I am able to show, and have learned a lot in the process. I hope this document can be a help to others designing similar robots, or using some of the same sensors or systems.

Appendix A: System Schematic



Appendix B: Source Code