Kickbot: A Spherical Autonomous Robot

Christopher Batten and David Wentzlaff
6.836 Final Project
{cbatten|wentzlaf}@mit.edu

1 Introduction

Kickbot is a custom built autonomous robot which wanders in its environment searching for people who will kick it. Kickbot's novel spherical body and solid construction make it ideal for kicking, and its carefully counterweighted central disk helps stabilize the robot after tumbling. Kickbot has a two part distributed control system which uses various sensors to avoid obstacles, detect when it is being kicked, and find people to antagonize.

There were three main objectives we hoped to achieve by building Kickbot. First, we wanted to build an autonomous robot from scratch. Too often it seems that building a robot simply means obtaining a pre-made robot control board, attaching a few sensors and four wheels, and then programming it to move around. We lacked the financial means to purchase a commercial control board with an advanced embedded microprocessor, but more importantly we wanted the experience of creating our own control system and experimenting with our own ideas for an interesting robot body. Part of what makes robotics so exciting is that it includes so many varied aspects: mechanical, electrical, behavioral, and control.

The second objective for building Kickbot is more specific. Most robots have strict orientation requirements such that they will not function correctly, if at all, when turned upside down or placed on their side. We wanted to create a robot where falling over was not a failure mode. Kickbot is specifically designed to correctly orient itself when knocked over or after bouncing into an obstacle. The third objective was to experiment with various emergent behaviors with a robot that we built ourselves. We were able to observe Kickbot exhibiting several interesting behaviors: both expected and unexpected.

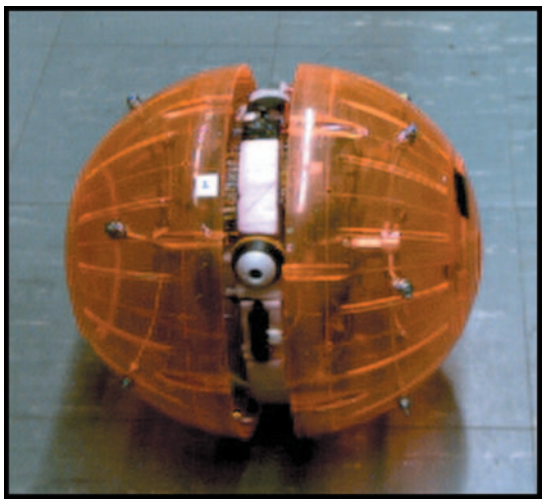


Figure 1: View of Kickbot

Overall, the project was a success. Kickbot is able to roll around an office building and avoid obstacles. Due to the specifics of its sensor placement, sometimes Kickbot does not see an obstacle and so it rolls into it, but Kickbot doesn't mind. It simply bounces off in a new direction. Kickbot can detect motion and rolls towards it in an attempt to find someone to kick it. When kicked it lets itself tumble, and when stable again Kickbot wanders off once more. But the experience we gained building Kickbot is just as important as the final robot itself. We learned how to machine our own parts, program a microcontroller, and build a robot body out of a hamster ball and some styrofoam. We learned the importance of decoupling capacitors, that things can and will break, and that sensors are perfectly happy to lie for no good reason. Ultimately, however, we learned that building a robot from scratch is great fun.

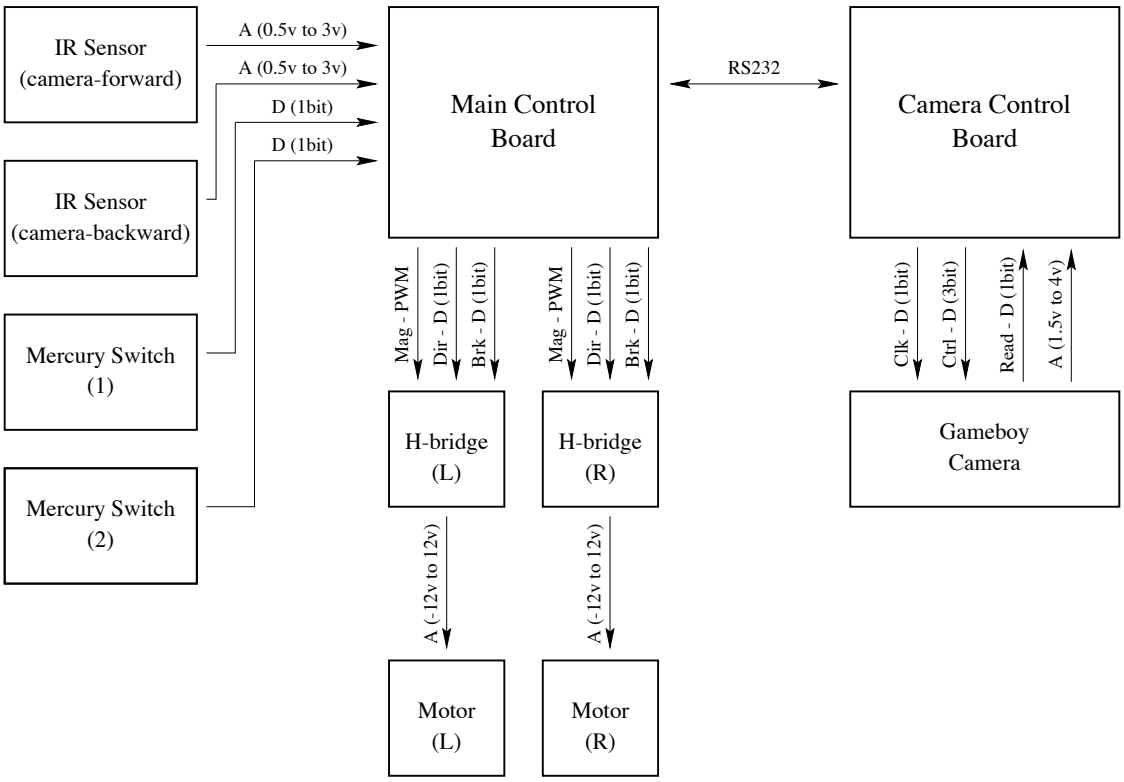


Figure 2: System-Level Block Diagram - Arrows indicate connectivity and format of communication between blocks. A = analog, D = digital, PWM = pulse width modulation.

1.1 System Overview

Kickbot is composed of two plastic half-spheres and a central disk (see Figure 1). The central disk contains all of the electronics, sensors, and motors and is attached to the half-spheres through an aluminum hub and six spokes. The central disk also includes significant counterweight to help stabilize the robot and permit forward motion.

Figure 2 is a system-level block diagram of the robot. The connectivity arrows also indicate the actual communication format: A indicates an analog signal and associated voltage range, D indicates a digital signal and associated number of bits, and PWM indicates a pulse width modulated signal. Kickbot has several sensors: two infrared range sensors mounted in opposite directions, two mercury switches to monitor orientation, and a small camera. There are two custom built control boards: a main control board and a camera control board. Each board has its own micro-controller and some additional logic (e.g. SRAMs, LEDs, dip switches). The main control board is responsible for monitoring the IR sensors and the two mercury switches, and also controls the two H-bridges which in turn control the two drive motors. The camera control board is responsible for operating the camera and image processing. Communication between the two control boards is through the standard RS232 protocol.

1.2 Outline

The remaining sections of this report will provide more details concerning the construction and operation of Kickbot. Section 2 discusses Kickbot's high-level behaviors within the context of a subsumption architecture. Section 3 concentrates on the mechanical aspects of Kickbot, while Sections 4 and 5 discuss the electrical and control aspects of the main control board and the camera board. Section 6 presents some final conclusions.

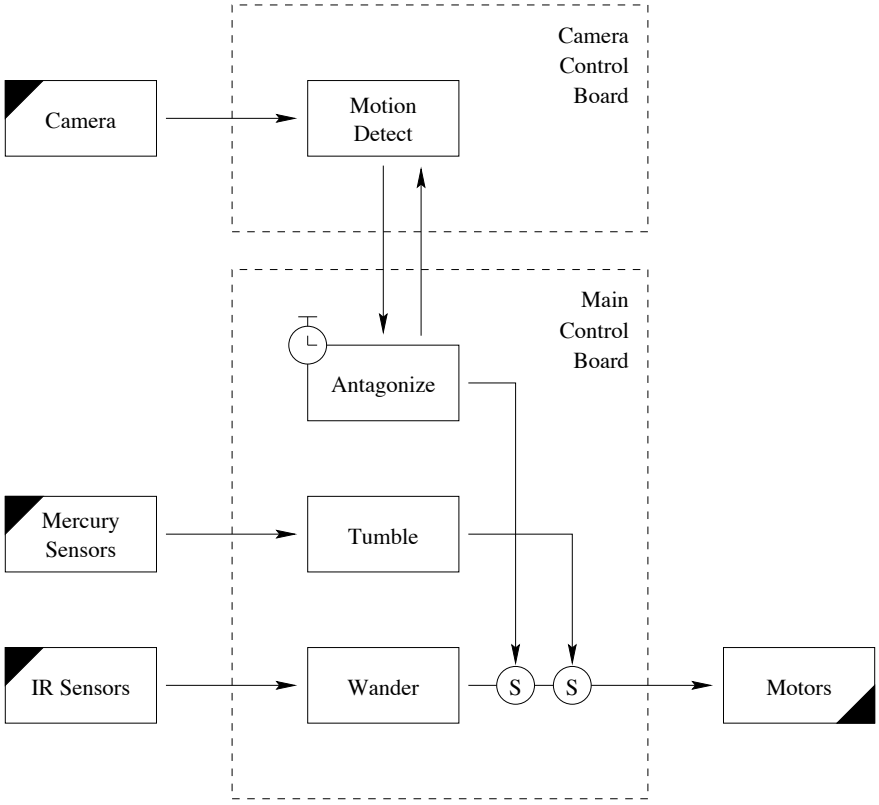


Figure 3: A Subsumption Architecture for Kickbot

2 Behaviors

To create Kickbot's high-level behavior we used a subsumption architecture which consists of three simple behaviors layered on top of each other: a wander behaviour, a tumble behaviour, and an antagonize behavior. Using a subsumption architecture was advantageous for three reasons: it helped us better conceptualize the robot's behavior, it allowed us to investigate emergent behavior at all three layers, and it was easily extended to a modular implementation. Figure 3 shows how these three simple behaviors are combined to form Kickbot's overall behavior.

System development initially focused on the wander behavior and then the other behaviors were added gradually. This incremental development is a key characteristic of subsumption architectures and definitely helped in managing the design complexity. Even the finished version of Kickbot exploits this modularity: there are several switches on the main control board which allow a user to choose between the wander behavior by itself, the wander and tumble behaviors, and all three behaviors. Although all three behaviors are implemented on the main control board, the majority of the work required for the antagonize behavior is actually implemented on the camera control board. This modularity made board and software development easier since once the interface between the two control boards was specified, we could work on each board independently. Each behavior is discussed in more detail below.

2.1 Wander

The wander behavior enables Kickbot to roll around its environment while avoiding obstacles. As shown in Figure 3, the wander behavior makes use of the two IR sensors to detect when an obstacle is in front of Kickbot. If such an obstacle is detected, then Kickbot should take some action to avoid the obstacle. Three different avoidance mechanisms were investigated: simple reverse, reverse and turn, and spin.

The *simple reverse* obstacle avoidance mech-

anism was actually an emergent behavior. An early version of Kickbot had very little counter-weight and was programmed to simply reverse directions when an obstacle was detected. It was expected that Kickbot would move back and forth between two obstacles. Kickbot's actual behavior was much more interesting and a prime example of emergence. Kickbot would see an obstacle and stop its motors, but because of the limited counter-weight, it would continue to roll into the obstacle. Kickbot would then bounce in an arbitrary direction and begin moving. The randomness of a complex environment allowed Kickbot to wander without any explicit turning mechanism.

Although the simple reverse technique was interesting, it was difficult to control and numerous collisions were straining the robot. The *reverse and turn* obstacle avoidance mechanism relies on an increased counter-weight to help Kickbot stop before colliding with the detected obstacle. Since Kickbot is symmetrical, it just moves in the opposite direction without needing to actually turn completely around. One of the motors starts in the reverse direction before the other motor, which causes Kickbot to turn slightly and head off in a new direction. The amount that Kickbot turns varies based on how charged the batteries are and the surface that Kickbot is moving on. Kickbot's small wheel base means that it turns much better on smoother surfaces such as tile. When moving on carpet, Kickbot turns significantly less if at all. This is a good example of situatedness, and it has consequences on Kickbot's overall behavior: Kickbot tends to explore more when moving on smoother surfaces than when moving on rougher surfaces.

The *spin* obstacle avoidance mechanism prevents Kickbot from becoming trapped between two obstacles. It was expected that Kickbot would never move into a situation where an obstacle was both in front and behind it, since the reverse and turn mechanism should enable Kickbot to avoid the obstacle completely. But the randomness and complexity of Kickbot's environment created several situations where it became trapped. To address this, the spin obstacle avoidance mechanism causes Kickbot to spin in place for some amount of time if an obstacle is detected both in front and behind. Kickbot then stops and checks to see if either direction is clear. If so, Kickbot proceeds in that direction, otherwise it spins again. This behavior significantly reduced the number of times Kickbot became trapped, but it also revealed another unexpected emergent behavior. Sometimes Kickbot goes over a small bump and the central disk accidently swings causing the forward IR sensor to point down. Kickbot sees the floor, thinks there is an obstacle, and stops quickly. If Kickbot is moving fast enough, this can cause Kickbot to swing in the opposite direction and the opposite IR sensors also sees the floor. Kickbot thinks it is trapped and spins in place. The result is that Kickbot occasionally spins in place and heads in a new direction even if no obstacles are nearby. This slight randomness causes Kickbot to explore in many new directions that it would otherwise miss if it always moved in a straight line.

2.2 Tumble

If Kickbot is kicked with just the wander behavior, its motors will spin wildly and Kickbot will not roll very far. The tumble behavior uses the two mercury switches to sense when Kickbot is being kicked, and then suppresses all motor messages with stop messages (as shown in Figure 3). This allows Kickbot to roll much more cleanly after being kicked and enables it to wait for a period time to make sure it has fully stabilized before continuing. From the observer's perspective, this stabilization period may make Kickbot appear to be catching its breath after tumbling before moving on.

2.3 Antagonize

The antagonize behavior can be layered on top of the wander and tumble behaviors to aid Kickbot in finding people to kick it. This behavior allows Kickbot to periodically stop and turn in a circle looking for movement using the camera. If it detects movement, Kickbot rolls towards the movement and bumps into the moving object. Kickbot tries to annoy the moving object so that it will be kicked. Figure 3 shows how

a timer on the main control board signals when Kickbot should start looking for movement. The camera control board handles all communication with the camera and does the necessary image processing to determine if there is movement in the field of view. The camera control board can also calculate if this movement is directly ahead, to the left, or to the right, and the main control board can use this information to turn slightly towards the movement.

The antagonize behavior is not a movement following behavior since Kickbot only looks for movement occasionally and must stop before capturing any images. Instead, the antagonize behavior enables Kickbot to focus more on areas in his environment where there is movement. This means Kickbot will have a better chance of finding a person to kick it.

3 Mechanical Design

Kickbot's mechanical design was primarily influenced by two key design constraints: falling over should not be a failure mode and Kickbot should be able to be kicked. To a lesser extent, finacial constraints also influenced Kickbot's mechanical design.

In the early days of the Kickbot design, Kickbot was going to be a cube with six symmetric drive sides. But unfortunately (or fortunately), we realized that having so many sides and drive mechanisms was prohibitively expensive. To overcome this problem while maintaining the robot's robustness to falling over, Kickbot was redesigned as a sphere. This novel spherical design consists of three parts: two half-spheres and a central disk. The two half-spheres rotate and the central disk provides a place to mount sensors and other electronics. The central disk is kept stable and upright with a counterweight on one-half of the central disk. The counterweight makes forward motion possible and helps reorient Kickbot when it is tumbling. For locomotion, Kickbot has two motors mounted on the central disk. Each motor is attached to one half-sphere and this provides a basic form of differential drive control.

3.1 Physics

To understand the physics of Kickbot, let's first assume that Kickbot's half-spheres are actually two large circular wheels attached to a central disk. Now let's assume that the wheels and central disk are equally weighted. Depending on the balance of rotational moment of inertia of the central disk, the rolling friction that the wheels experience with the floor, and the rotational moment of inertia of the wheels, the central disk would probably start to spin in the center of the robot and possibly some small forward locomotion would occur. But what if we increase the weight of the central disk?

If the mass of the central disk is increased evenly across its volume, then this would increase the rotational moment of inertia of the central disk. A heavier central disk increases its resistance to rotation and makes the outer half-spheres rotate more. The result is forward motion.

But in addition to forward motion, Kickbot also needs a stable platform for its sensors. To make the central disk more stable, we can place a counterweight such that one-half of the central disk is significantly heavier than the other half. This asymmetric weight distribution is also useful in propulsion of the robot. As the robot raises its counterweight above its rest point, the weight is moved in front of the location that the robot contacts the ground and the robot falls forward. A closer analysis can be see in Figure 4. In this figure, the circle represents the half-spheres from a side view and the counter-weight is shown in its rest position. The large X denotes the center of gravity of the robot system. Figure 5 shows the system after the motors have tried to drive it forward a bit. As can be seen, the center of gravity is in front of the contact location with the ground. There is now non-zero net torqoue and the robot falls/rolls forward.

3.2 Construction

In addition to acting as Kickbot's wheels, the half-spheres form a protective shell around Kickbot. The half-spheres must be strong and sturdy to withstand kicking, but not so heavy that they

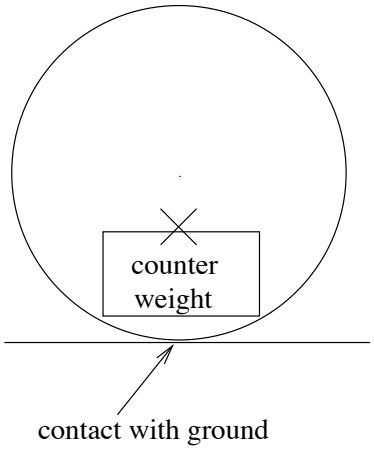


Figure 4: Kickbot in Rest State

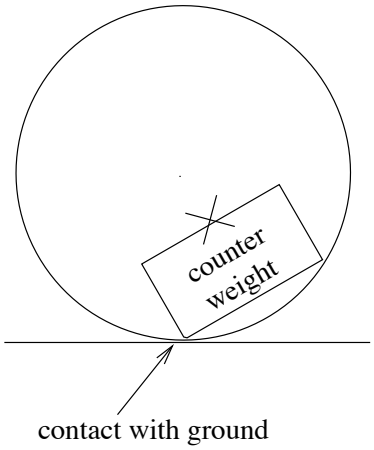


Figure 5: Kickbot Falling Forward

outweigh the central disk. We found the perfect solution at our local pet store: an 11 inch diameter hamster ball. Hamster balls are made out of a flexible yet durable plastic. This flexibility turned out to be very useful, since it acted as a simple shock absorber during collisions.

Attaching the half-spheres to the motor axels turned out to be much more challenging than we first expected. It was difficult to design a mechanical connection between the curved surface of the sphere and the motor axel that would be both strong and relatively lightweight. We eventually chose a custom hub and spoke system, where the hub attaches to the motor axel and six spokes connect the hub with the half-sphere. The hub was machined from stock aluminum and has six threaded holes around its perimeter for

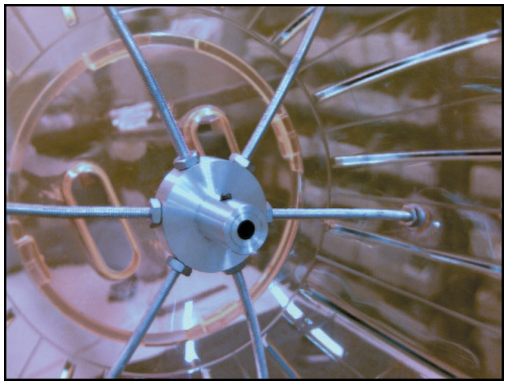


Figure 6: Detail of Hub and Spokes

the spokes.[1] A close-up of the custom machined hub can be seen in in Figure 6. The hub has a hole in it for the motor shaft and a set screw to hold it onto the motor shaft. Threaded rods were inserted in to the perimeter holes to act as spokes. The threaded rods were bent to meet the half-spheres at a 90° angle and are connected with bolts to the hamster ball.

A consquence of a spherical body is that Kickbot has a wheel base of only 3cm. This is not a problem when moving Kickbot forward, but can create a significant problem when trying to turn Kickbot with a speed differntial between the two motors. Kickbot has trouble spinning in place on carpet, even with both motors at max speed in opposite directions. Kickbot is much better at spinning on smoother surfaces such as tile. We attempted to expand the wheel base with strips of rubber mounted to the half-spheres, and this did slightly increase Kickbot's ability to turn quickly on carpet. A better solution would be to user drive motors with a higher gear ratio and thus more torque.

Kickbot's central disk is a circle cut from pink foam house insulation with a diameter slightly smaller than that of the half-spheres. We decided to use foam because it is relatively stiff, easy to cut with a hot wire, and lightweight. A lightweight central disk makes forming the asymetric weight distribution mentioned in the

[1]Machining the hub was quite a challenge since neither of the authors had much experience with this type of work. We would like to thank Fred Cote who runs the Edgerton Center Student Machine Shop for helping us learn the basics of machining so that we could build our part.

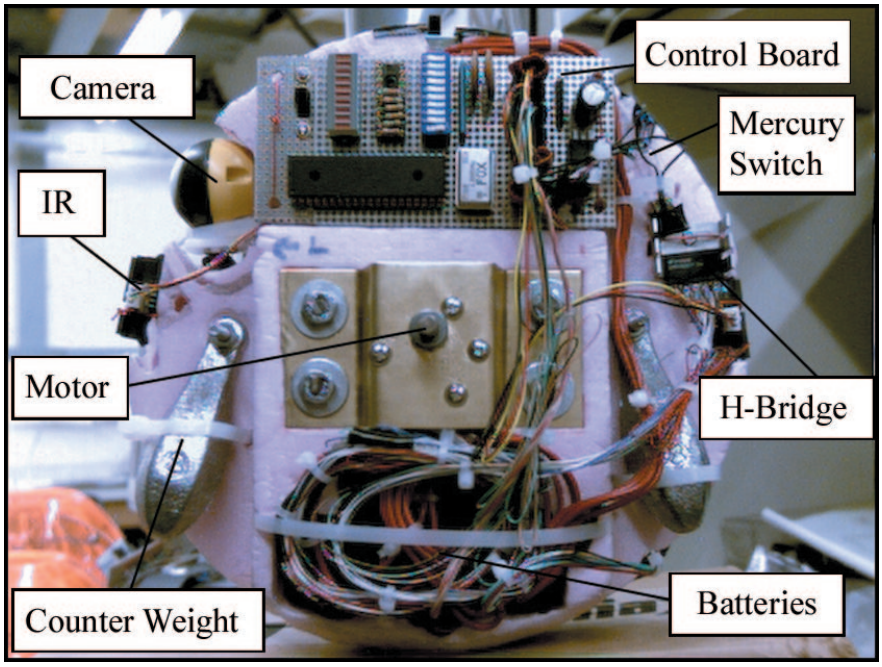


Figure 7: Detail of Main Control Board Side

previous section much easier. The drive motors are mounted with custom brass plates such that their shafts are directly in the center of the central disk and are perpendicular to the disk.

Many things are attached to the central disk with both wire ties and bolts. Lead weights and batteries act as the counterweight and are therefore placed below the motor shafts. The IR sensors are mounted on opposite sides of the central disk and the camera is mounted directly above one of the IR sensors. Figures 7, 8, and 9 show how things are mounted to the central disk.

4 Main Control Board

This chapter describes the main control board in more detail. As mentioned in Section 1, the main control board is responsible for monitoring the IR sensors and the mercury sensors, and for controlling the drive motors. The main control board also acts as the master for the camera control board, since it periodically asks the camera control board to perform some work on its behalf.

4.1 Board Description

The primary component on the main control board is a PIC16F877 micro-controller. This micro-controller was selected since it includes an integrated analog to digital converter, USART controller, and PWM generator. A very simple programmer was constructed using a resistor, a parallel cable, and a host computer running Linux. The PIC16F877 is capable of in-circuit low-voltage programming which made it very convenient to work with.

Several additional components are also on the main controller board. Figure 10 shows the main control board and labels the various board components. A 4MHz oscillator is used to clock the micro-controller and a slide switch is used to choose between programming and running mode. Standard AA batteries were used to supply power to the main control board and the camera control board. The board requires 5V, but since one AA battery provides 1.5V, a voltage regulator was needed to level convert 6V (4 AA batteries). We used a Texas Instrument 7805 voltage regulator for this purpose. Although the specifications list 6V as an acceptable input voltage, we found the output voltage to be unacceptably less than 5V unless the input voltage was

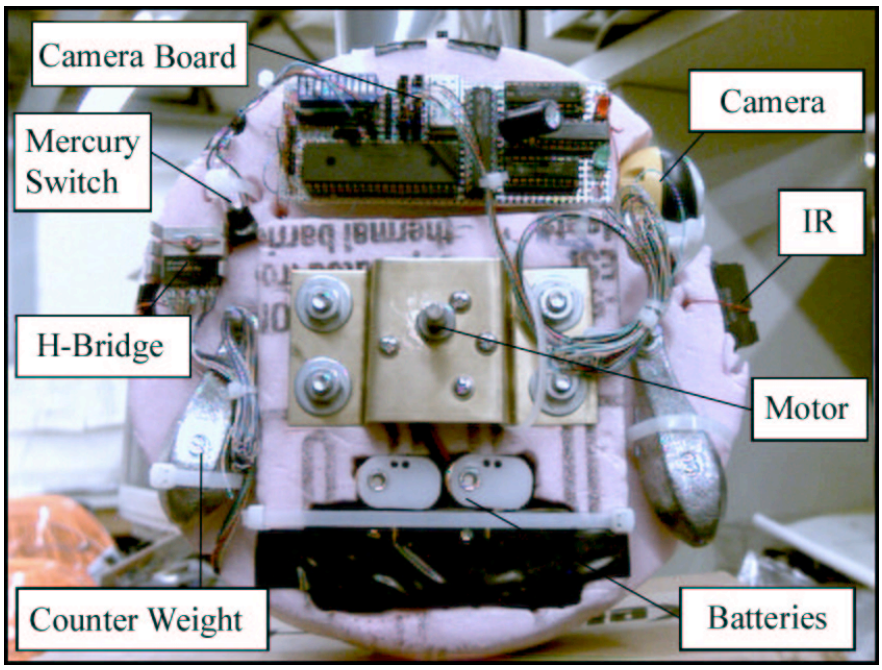


Figure 8: Detail of Camera Control Board Side

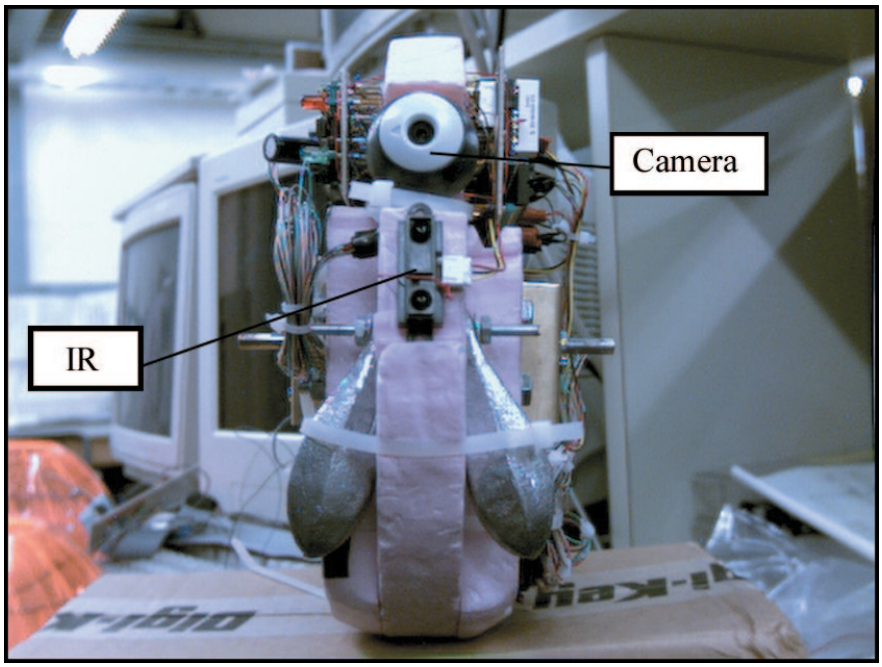


Figure 9: Detail of Front

set to 8V or more. We therefore added two more AA batteries to bring the total number of AA batteries supplying the controller boards to six. Note that the drive motors use a completely different set of AA batteries to avoid current spikes when the motors switch directions or encounter resistance.

A set of ten LEDs provide a way for the micro-controller to visually display status information. The current version of the main controller board software uses one LED to indicate when the robot is in tumbling mode, and a second LED to indicate when the robot is in antagonize mode. The remaining eight LEDs are used to display the input from the currently active IR sensors. Six dip switches are available for configuration. The current software version uses one of these switches as a board enable, a second switch to enable the tumbling behavior, and a third switch to enable the antagonize behavior.

The main controller board includes nine ports: one 4 pin port for power, two 4 pin input ports for the IR sensors, two 2 pin input ports for the mercury switches, two 4 pin output ports for the motor control, one 5 pin input port for the programmer, and one 4 pin port for the interface to the camera control board. We chose to use Sharp GP2D12 infrared detectors which claim a range of 80cm, although our testing found the range to be closer to 60cm. These IR sensors are wired to the A/D inputs on the micro-controller. An external voltage divider is used to provide a voltage reference of 3v to the A/D converter, since the IR sensor output is only 0.5v to 3v. The mercury switches and motor control bits are wired to the digital I/O pins on the micro-controller. The mercury switches are mounted at opposite angles such that both switches will be on only if the robot is not more than 90° away from its stable vertical position. Using two mercury sensors instead of one provides much more flexibility in setting this angle.

The main control board communicates to the camera control board using the standard RS232 protocol and uses the integrated USART interface on the micro-controller. Using the RS232 protocol was particularly helpful since it only uses two micro-controller pins and saved us the trouble of developing our own custom interface.

4.2 Drive Motors

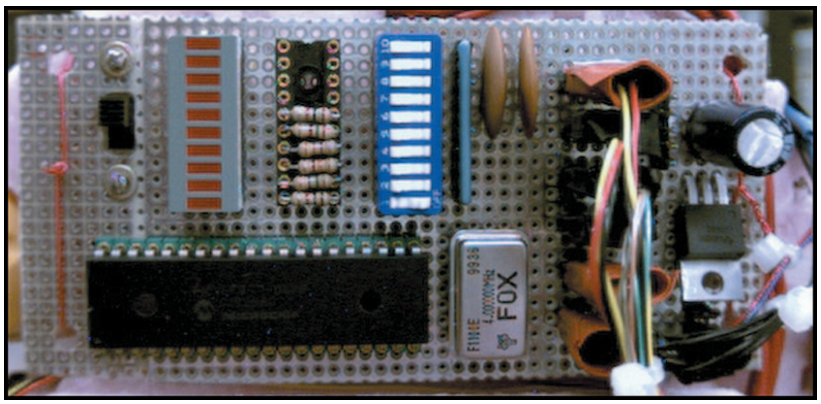
National Semiconductor LMD18201 H-bridges were used to connect the micro-controller to the drive motors. These H-bridges take a pulse width modulated input for the motor speed and two digital inputs to indicate direction and brake. The PIC16F877 includes two PWM channels that were directly connected to the H-bridges. The H-bridges provide a $\pm$12V signal to the drive motors. Ten AA batteries supply the H-bridge with 15V, which the H-bridge then reduces to the appropriate voltage based on the PWM signal.

Two 12V DC 200rpm motors were chosen as Kickbot's drive motors. These motors have a 30:1 gear ratio and can provide significant torque. Even so, we found that more torque would have been helpful when turning on rough surfaces (as discussed in Chapter 3).

4.3 Software

Several hundred lines of assembly code were written for the micro-controller on the main control board. It was definitely a learning experience to program in a very primitive assembly language. The control code consists of a large main loop which first performs the A/D conversion to determine the current IR sensor value. If this value exceeds a threshold, an obstacle has been detected and the control software jumps to the corresponding obstacle avoidance code. The main loop also checks the mercury switches and if either switch is off, then the control software enters the tumbling code.

The timer for motion detection is implemented with a simple nine bit counter which starts at all ones and is decremented every iteration of the main loop. This results in approximately 30 to 40 seconds between stops to detect motion. The time interval varies based on what the robot is actually doing (e.g. avoiding obstacles increases the time per main loop iteration). As mentioned earlier, communication with the camera board uses a standard RS232 protocol. When the main control board wants the camera board to perform motion detection it sends a DETECT message. The main control board should make



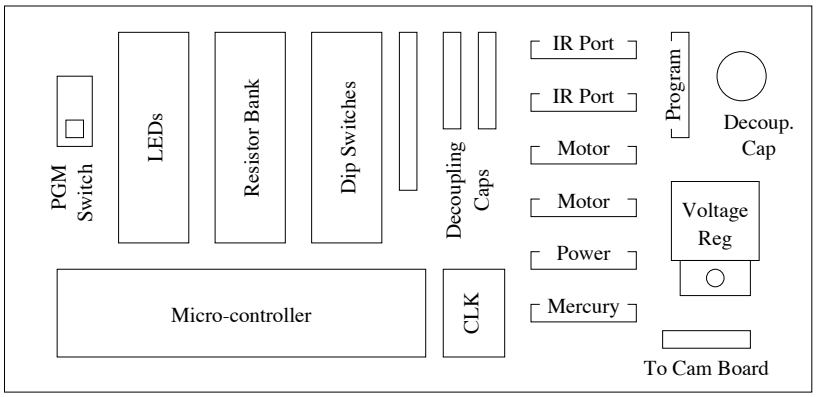
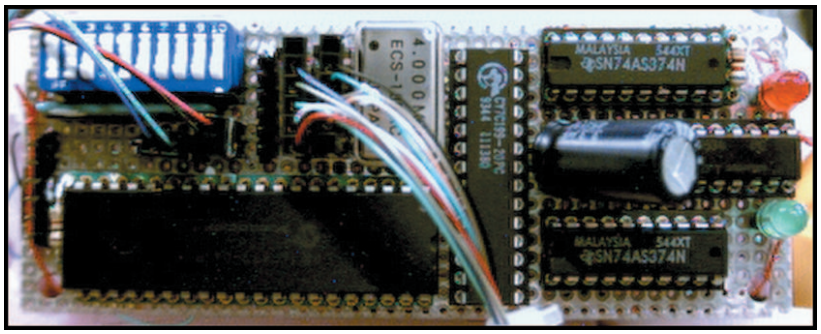


Figure 10: The Main Control Board



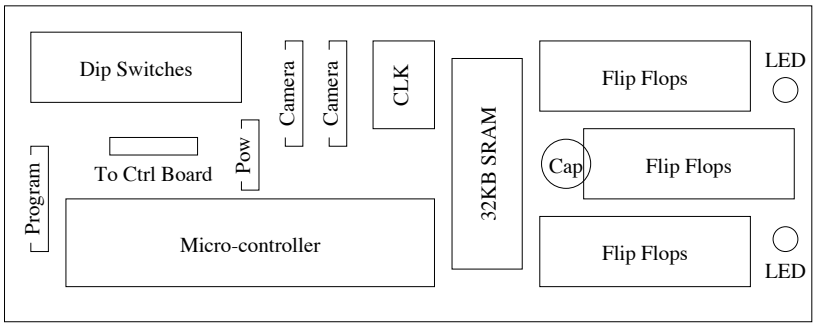


Figure 11: The Camera Control Board

sure that Kickbot is stable before sending such a message. The camera control board will then reply once it has captured the images and performed the necessary image processing. The control board communication messages are listed in the following table (where MCB stands for main control board and CCB stands for camera control board).

Direction	Message	
MCB → CCB	DETECT	0x01
CCB → MCB	NO_MOTION	0x02
CCB → MCB	MOTION_LEFT	0x03
CCB → MCB	MOTION_FOR	0x04
CCB → MCB	MOTION_RIGHT	0x05

5 Camera Control Board

The camera control board is responsible for capturing images using Kickbot's camera and then processing those images to aid Kickbot in finding people to kick it.

5.1 Hardware

It was realized early during this project that the IR sensors that Kickbot has would not be adequate to detect humans, thus a different sensor was needed for Kickbot. Unfortunately, Kickbot's processing capabilities are very low so it is not able to process the output of (nor does it need) a very high quality image sensor. The image sensor that was chosen was the Gameboy Camera. This was an inexpensive ($10 on eBay) gray-scale camera with 128x128 pixel resolution.

The camera control board is composed of a PIC16F877 micro-controller, 32KB SRAM frame buffer, and some TTL flip flops which allow for pin multiplexing off of the PIC. The PIC has an on-board analog to digital converter which was very helpful because the Gameboy Camera is actually a CMOS sensor which supplies an analog voltage serially to output a picture.

5.1.1 Gameboy Camera

Calling the camera a "Gameboy Camera" is a little misleading. In actuality, to use the camera, you need to open the Gameboy cartridge and remove everything having to do with the Gameboy and instead directly connect to the imaging chip. The imaging chip is a Mitsubishi M64282 Artificial Retina CMOS sensor, and it includes the ability to do hardware edge detection.

The interface with this camera is quite complicated. The interface boils down to a bit serial control channel going to the camera, and analog output which serially communicates the CMOS sensor's data out. Inside of the camera chip there is a small register file which you use to set up certain functions of the chip such as exposure time, positive vs. negative image, edge detected or not, and gain. To load these registers you bit serially clock in a data and address to the camera, and then raise the LOAD line to load the register. After all control registers have been setup, you toggle the START line and provide a clock. After a certain amount of time the camera raises its only digital output, READ, which tells you that data is ready and that you should start to capture data. Now for the next 128x128 clock transitions you sample the Vout (analog pixel out) and store it. The analog output swings from about 1.5V to 3.5V.

Effectively using this camera was difficult because of this interesting protocol and the stringent timing constraints on the A/D when capturing images. The actual frame rate of this camera is a function of how fast your A/D samples and how much light is in a scene. This intuitively makes sense because if there is more light, you need to expose the sensor for a shorter period of time and thus can take pictures more quickly.

5.1.2 Board Description

To connect to the Gameboy Camera, a unique interface board was needed that was capable of doing quick (tens of micro-seconds) A/D conversion, have some place to store images, have some processing power to process the image, and have communication with the outside world. The board made to fulfill these needs contains a PIC16F877 which is a 4MHz 8 bit accumulator based micro-controller with only 300 bytes of ram (Figure 11. The small ammount of onboard RAM posed a problem as a single frame from the camera was 16KB. To solve this, a 32KB SRAM was hooked up to the general use digital

I/O pins on the PIC. External TTL flip flops enabled the PIC to time multiplex the address and data buses ont the same PIC digital I/O pins.

To obtain pictures, the PIC's on-chip A/D was used. To get good contrast in the images a voltage reference needed to be provided to the A/D. A voltage divider was used to generate the 1.5V V_{ref-} and 3.5V V_{ref+} needed. The on-chip A/D is limited to 24 micro-seconds, thus if there were no other overheads, one can only capture 3 frames a second ($24\mu s * 128 * 128 = .39s$).

The camera control board also needs to communicate gathered information with the outside world. To that end, the PIC's on-chip UART was used. We used a 9600 baud connection to communicate pictures and information to the outside world. The same RS232 interface is used to communicate with an desktop computer for debugging and with the main controller board.

5.2 Software

There were two main software interfacing issues. One being how to use serial communications. Interfacing with RS232 was surprisingly easy. On the PIC all you really have to do is set up some special purpose registers which set the baud rate. Then to send and receive you simply write to a send register and read from a recv register. You can also check a different register to see if incoming traffic has come thus allowing you to do a blocking receive. A problem which cropped up in the course of development was getting the two controller boards which were started up out of sync into sync via this asynchronous communications channel. It was made worse by the channel being full duplex. Having a full duplex channel is usually a good thing, but getting to a point where two PICs can have a conversation requires a synchronizing handshake. A three way handshake was used which allowed the two PICs to get synchronized after startup. After that the interfacing protocol works by having the motor control board ask if the camera board saw motion and then the camera control board responds saying if any was detected and where.

The other interesting piece of interfacing software is the software which communicates with the camera. This software implements the protocol described in Section 5.1.1. This software required the use of the PIC's interrupt facilities to generate a stable software clock for the camera device. This was an unfortunate artifact of the camera chip using the same clock pin for logic control and image capture. When the PIC is configuring the registers on the camera it slowly clocks in data, and when an actual image is being captured, an interrupt routine gets fired once every 48 micro-seconds to toggle the clock pin. If the the camera is outputting pixels, it analog to digital converts them and stores them to the off chip SRAM.

5.2.1 Picture Capture

The only real feedback to know if you have properly configured the camera is to take a look at pictures generated by the camera. To do this, PIC assembly was written to capture a frame and then dump the frame to the computer over the RS232 port. On the computer side, a terminal program waits and receives the data into a file. This file is post-processed in Matlab, and then displayed as an image. Figure 12 shows the robot looking at David's legs in the lab. Figure 13 shows Kickbot exploring Tech Square's 6th floor playroom. Lastly Figure 14 shows Kickbot's view of a hallway on the 6th floor.

5.2.2 Motion Detection

In order for Kickbot to find people to antagonize, it needs some way to detect them, and an easy way to detect people is to look for motion. To detect motion, Kickbot simply captures two consecutive frames and it unsigned subtracts one frame from the other. Any motion will show up as a difference in the two frames. This of course assumes that the camera is not moving. The test version of the program takes two frames, subtracts one from the other, and then outputs them to the serial port. An example of the results can be seen in Figure 15. This is a photo taken with Chris on the left side of the frame moving himself and his hand. Figure 16 shows the motion map (the diff done on the PIC) of the two captured frames. The white represents the movement and indeed it picks up Chris moving

Figure 12: Kickbot Looking at Dave's Legs

Figure 13: View of 6th Floor Playroom

Figure 14: View Down 6th Floor Hall

Figure 15: Picture with Chris Moving

Figure 16: Chris Moving Motion Map

and his hand moving.

For Kickbot, a program analyzes the difference between two frames and reports back to the main controller board if there was motion and if so was the motion to the left, right, or straight ahead. This communication occurs over the RS232 port between the two boards. An optimization was made that does not require the PIC to write the picture back to the SRAM, but does a running summation of how much movement is detected into the three buckets, left, right, and middle. Lastly these totals are compared to a hard-coded movement threshold which is tuned for motion detection.

6 Conclusion

Overall we feel that Kickbot was a great success. We fulfilled our objectives of making a real robot that you can kick and used it to investigate interesting real world embodied behaviors.

From this project we are able to draw three conclusions about building real robots. First, building a real robot is *expensive*. We received no support from our research groups and had to finance the project ourselves. Second, building a real robot is *hard*. We had to learn how to machine custom parts, and real world considerations such as power supply noise, faulty wiring, IR sensor noise, and the physics of a spherical robot can cause significant complications. Finally, building a real robot is *fun*. We enjoyed this project and would build another robot in the future.