Engineering Senior Design

MuLIR – Multi-Legged Interactive Robot

Pat Mills
Jay Herrick

November 11, 1996

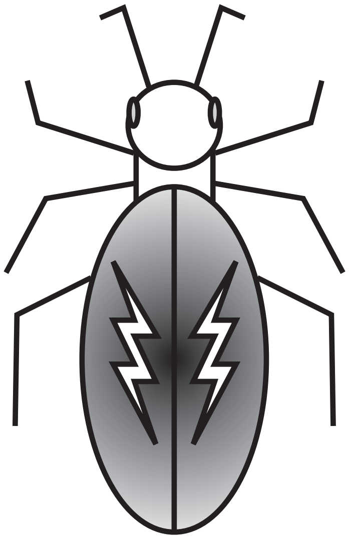


Table of Contents

List of Figures and Tables

Abstract

This report explores the intricacies involved in designing and implementing a multi-legged interactive robot (MuLIR). Motivation to design such a robot came from exposure to similar projects developed by the Mobile Robot Group at MIT's Artificial Intelligence Laboratory.

Several goals were set before design began on MuLIR, these are listed in order of importance:

1.) Design and Implementation must be possible in one semester.
2.) The final design must have six legs each with two degrees of freedom and capable of variable speeds.
3.) The robot must be self-contained.
4a.) The robot must interact quickly with its environment.
4b.) The robot's control system must exhibit actions indicative of "higher intelligence."
5.) The control system should be fault-tolerant.
6a.) The system should be easily modified to allow testing of new control systems and actions.
6b.) The hardware should be easily extensible to support new motions and sensors.

MuLIR's control system is based on five Motorola M68HC811E2 microprocessors operating in parallel. Three of the processors are dedicated to controlling the legs (each processor controls a left / right pair of legs), one to sensory input, and one for determining MuLIR's "higher reasoning" and communicating with a host computer for code changes and diagnostics. The control system is based on a new model of behavior called subsumption architecture. Commands for movement are generated by the control processor based on feedback provided by the leg processors and information about the environment provided by the sensory processor. Each command is sent over a serial link and is received by all processors, this allows for actions to occur in parallel.

Although time constraints prevented observation of MuLIR in full walking mode, some interesting results were gained. Throughout the design process prototype legs were created to ensure that the design goals would be met. Once the final prototype had been approved, it was used to test the leg software. Once the leg software was perfected, the prototype and software were used to simulate actual walking patterns. The interprocessor communication software was also tested using several M68HC11EVB boards. However, at the time of this printing it is unknown whether the integrated system will operate as a whole. It is possible to make some educated predictions about performance; the assembled unit is capable of supporting and lifting its own weight. Again, using the

M68HC11EVB board it is possible to test the legs in simple patterns to see whether the robot will be able to lift its weight from a resting position and to see whether forward motion is possible.

High expectations were set for this project, and even the basic results achieved justify the cost of development in both time and money. MuLIR will provide a test bed for future robotics projects whether they focus on mechanical, electrical, or computer engineering. The hardware and software developed are highly modular and easily extendible. An unexpected result of this project was the enjoyment experienced as the project neared completion. MuLIR proved fun to play with and satisfied many childhood dreams. It is the authors' hopes that anyone interested in robotics will be able to use MuLIR to explore their own dreams.

Introduction

Mobile robots have wandered the halls of universities and R&D institutions for years. Until recently, these robots have been developed with a top-down reasoning strategy. This traditional approach is based on world modeling and planning. In order for a robot to move, it must know its location and state as well as the location and state of all other objects in the world. Based on this information, the robot plans a series of actions that achieve the intended motion. World modeling has great seductiveness for researchers because of the guarantees and optimizations that are possible. Routines exist that will guarantee that the generated plan will accomplish the task or prove that the task is impossible. Furthermore, a plan may be optimized before execution. However, such plans require vast computational resources and time. For these reasons, most traditional robots have been large and cumbersome, often depending on external computers to perform complex calculations. Perhaps the biggest problem with top-down reasoning is the amount of time required to compute even simple plans. If the world changes from when planning is initiated and when it is carried out, the plan may fail. These time restraints led researchers to limit the robot's environment, but even then motion at speeds of one foot per hour was not uncommon, and interactive robots were either cost prohibitive or impossible.

A more biologically feasible solution is bottom-up programming. With this approach, sensorimotor skills replace higher-level thought processes as the basis for intelligence. Inspiration came from studying insects, and how they navigate in real-time in complex environments. For example, a fly buzzing around a room does not know whose room it is in, where the exits are located, or what types of furniture are in the room. It merely acts based on sensory inputs such as heat, odor, sound, light intensity, and color. A fly does not have time to compute a complex plan for object avoidance while flying. Researchers attempting to model insects' ability to quickly adjust to environmental change developed a model called *subsumption architecture*.

Subsumption architecture was developed by Dr. Rodney Brooks and the Mobile Robots Group at the MIT Artificial Intelligence Laboratory. Since its introduction in the late 1980's, subsumption architecture has been steadily gaining support in the robotics world. Subsumption architecture combines distributed real-time control with sensor-triggered behaviors. Instead of judging the reliability of sensor readings, sensors are dealt with implicitly in that they initiate behaviors. Since sensing is so closely linked to actuation, behaviors can be viewed as simple reflexes. This intuitive leap means that subsumption requires no world model and therefore is computationally simple.

Behaviors are layers of control systems with lower-level real-time behaviors (such as "avoid objects") on the bottom and higher-level goal-oriented behaviors (such as "track object") on the top. Appropriate behaviors are activated by sensors and run in parallel. This allows conflicting sensor readings to be passed on as conflicting behaviors. Conflicting behaviors are then resolved by subsumption. For a given state, a dominant behavior is determined using prioritized arbitration. If two behaviors conflict, then the higher-level behavior subsumes the lower-level, otherwise, both behaviors occur (called *behavior fusion*).[1]

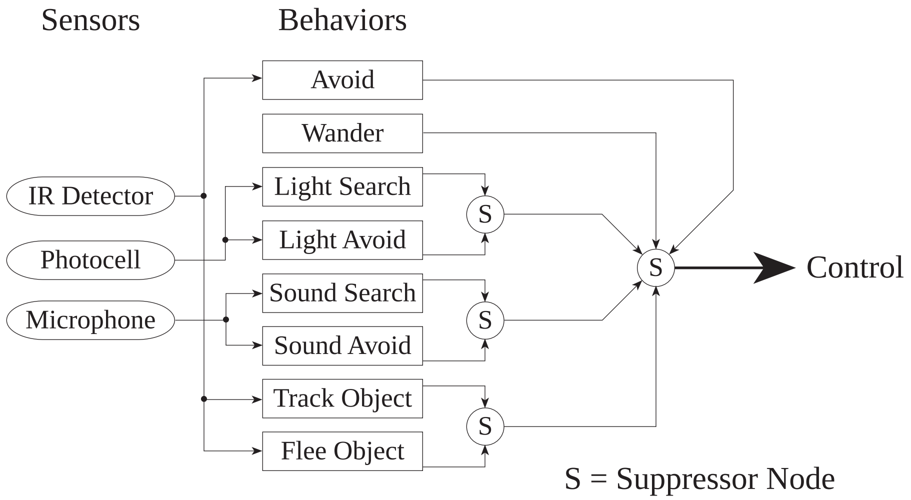


Figure 1: Subsumption Architecture with Behavior Fusion

MuLIR was designed to imitate an insect as closely as possible. It is hoped that insight into interactive robotics will be gained by careful analysis ofMuLIR's behaviors. As behaviors are added, complex new behaviors emerge; these new behaviors are the results of behavior fusion or the combination of lower-level behaviors into more complex parallel higher-level behaviors. Analysis of how insects interact with their environment provides a good starting point for robotic behaviors.

The design for MuLIR was guided by a simple set of desired behaviors:

Behavior 1.) Avoid Collisions
Behavior 2.) Wander
Behavior 3.) Track Light / Hide
Behavior 4.) Track Sound / Hide
Behavior 5.) Track Object / Flee

[1] Jones. pp. 243 - 251.

In order to realize these behaviors, processors were laid out so that response time to sensory input would be minimized. Each leg processor controls a left / right pair of legs. The processor waits for commands from the control processor or interrupts generated by movement. Sensory input is received and analyzed by a dedicated sensory processor. The control processor receives state information from the leg processor and sensory processor and issues commands based on active behaviors.

The rest of this report explores the intricacies involved in designing and implementing a walking mobile robot. The robot's basic design from hardware to software impacts resultant behavior; therefore, careful analysis of the design process and the reasons for design choices will be discussed. After design, the theory behind the microprocessors and the software they run is discussed. Finally, implementation issues will be addressed and results analyzed.

Design

When this project was beginning, several goals were set. These goals guided the decisions that yielded MuLIR on both the hardware and software levels.

1.) Design and Implementation must be possible in one semester.
2.) The final design must have six legs each with two degrees of freedom and capable of variable speeds.
3.) The robot must be self-contained.
4a.) The robot must interact quickly with its environment.
4b.) The robot's control system must exhibit actions indicative of "higher intelligence."
5.) The control system should be fault-tolerant.
6a.) The system should be easily modified to allow testing of new control systems and actions.
6b.) The hardware should be easily extensible to support new motions and sensors.

Furthermore, several desired behaviors were also defined:

1.) Avoid Collisions
2.) Wander
3.) Track Light / Hide
4.) Track Sound / Hide
5.) Track Object / Flee

Hardware:

It was determined that hardware design should proceed in the following order: legs, chassis, sensors, and finally chip layout and wiring. Leg design began in late September, 1993, although research began as early as the spring of 1992. Beyond the goals set for the overall system, the following goals / requirements were set for leg design:

1.) Simple modular design (easily replicated).
2.) Quick and easy access to parts.
3.) Variable speed and good top speed.
4.) High torque, the leg must be capable of supporting and moving the robot's weight.
5.) As low current draw as possible especially when leg is not in motion.
6.) Small, compact, and light.

The original leg design was based on a cylindrical telescoping leg which was capable of contracting and expanding along its primary length. This length change enabled the leg to be easily lifted and moved forward. The leg was designed in three parts: an outer shell which attached to a motor mounted on the main body, an inner shell which moved up and down in channels set inside the outer shell, and a fixed protraction which provided the leg's major length.

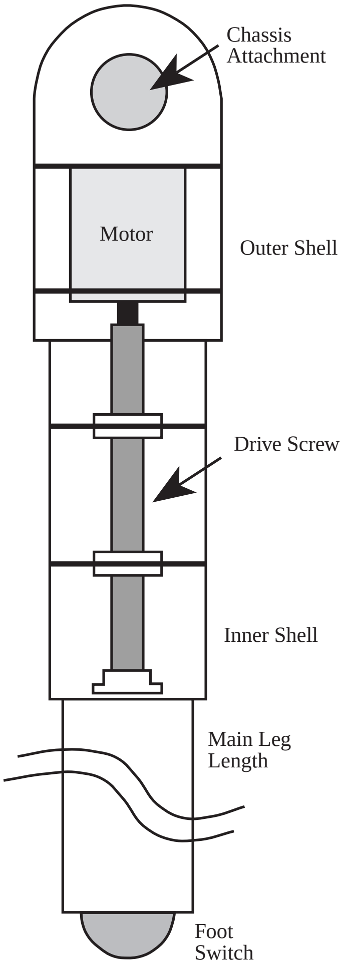


Figure 2: Telescoping Leg Design

Forward motion was achieved by contracting the leg until a sensor indicated full contraction had occurred, the leg was then rotated from the base so that it pointed slightly forwards, finally the leg was expanded until it reached maximum expansion or the foot sensor indicated that the leg was on the ground. This design was chosen since it met the defined goals and minimized the hardware and software required for walking. However, once a prototype was built, it was discovered that the leg tended to become stuck in either the compressed or expanded positions due to friction on the screw contacts. Also, attaching the limit switches in the outer shell was difficult.

At this point in the design, new leg designs were considered. One of the more simple solutions is shown below, but was rejected since it allows only one degree of freedom and makes multiple gaits difficult.

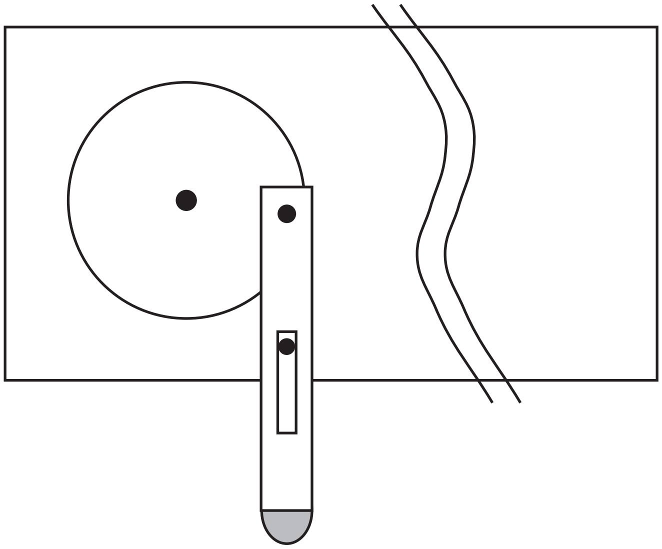


Figure 3: Simple Leg Design

In early January 1994, another satisfactory design was proposed. This design is quite similar to the designs used by Genghis and Attila (two six legged robots developed at MIT). The design is more complex than previous designs (yet simpler to machine), but allows greater flexibility and control.

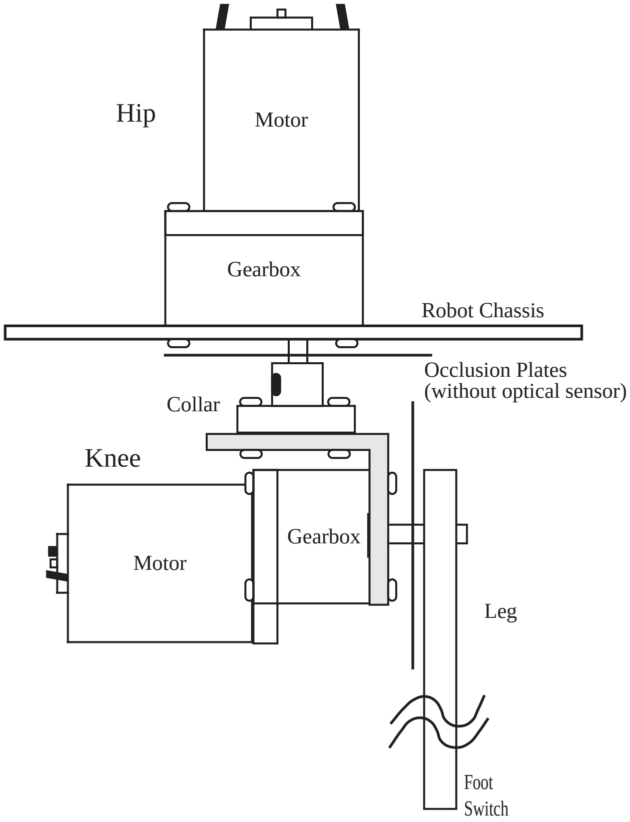


Figure 4: Early Hip / Knee Leg Design

This leg design consists of one DC gearhead motor (the hip) with its flattened drive shaft protruding vertically down from the chassis. This motor swings the leg in an arc from front to back, providing forward / backward motion similar to the telescoping leg. An identical motor (the knee) is fixed to the shaft of the first, with its shaft horizontal and parallel to the side of the robot. This motor swings the leg up / down in an arc out to the side of the robot. The leg itself consists of a solid shaft fixed by a set screw to the shaft of the knee motor. A foot switch at the end of the leg provides useful feedback. A complete forward cycle consists of a coordinated effort between the hip and knee. First, the leg is raised via the knee motor. Then the leg assembly is rotated forwards by the hip motor. Finally, the leg is lowered by the knee motor until it reaches a lower limit or touches the floor.

Early attempts at machining this leg were only minimally successful due to the space constraints imposed by the feedback mechanisms. It was deemed desirable to provide feedback with potentiometers since they provide continuous information which can then be discretely sampled via an A/D converter. However, no suitable potentiometers could be found that would easily interface with the chosen motors. A decision was made to keep the motors due to the high cost of replacing them; furthermore, the motors already in stock had low current demands and high torque. Infrared optical sensors were chosen to replace the potentiometers, one pair on the knee and one pair on the hip. They provided limit information and location at discrete points. The following is the occlusion plate used in the optical sensor:

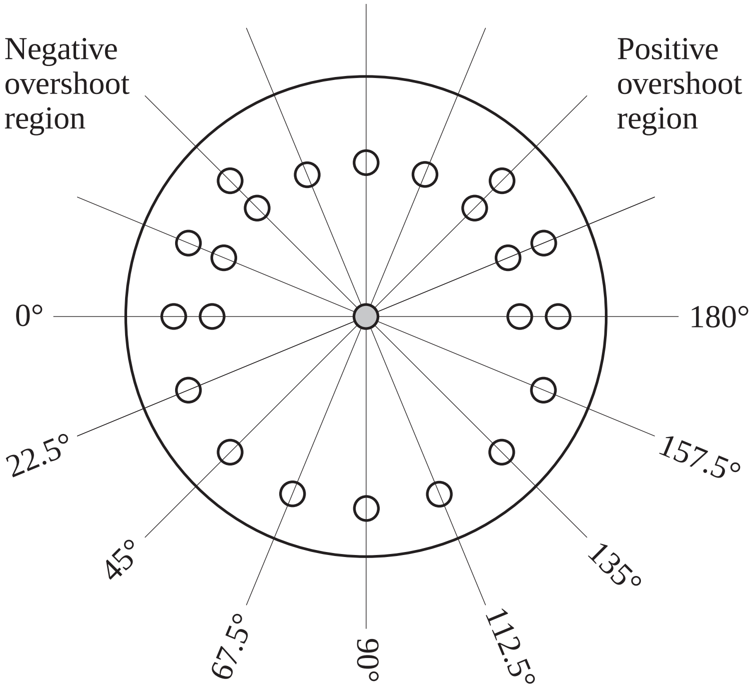


Figure 5: Early Occlusion Plate

Again, problems occurred during machining. The sensor plates were too complicated and large to be integrated into the leg design. To simplify the design, the feedback loop from the knee was replaced with two limit switches (spring-loaded break-before-make SPST), and the feedback loop from the hip was replaced with a simple infrared optical sensor and one double-throw limit switch (spring-loaded break-before-make SPDT). The resulting leg design is open-loop designwhen raising the leg; this was deemed acceptable since it is only important to know if the leg is up, down, or has touched the floor (provided by the floor switch).

Other problems with this design involved attaching the hip to the knee. The motors have a flat face with three 6-32 tapped holes for mounting (the shaft protrudes from the gearbox normal to this face). The early designs had a plate mounted to the face of the knee motor which extended up, bent 90°, and bolted to a collar over the gearbox. The collar then attached via a set screw to the shaft of the hip motor. While a collar was found for an early prototype, it proved difficult to find enough of the exact size required.

A new design was proposed early in February 1994. The basic leg design was kept, but the simplified feedback system was used, and a flat plate was introduced for mounting the hip motor to the knee motor. The plate has a groove in the side which is mounted against the gearbox. To connect the two motors, the shaft of the hip motor is pinched between the plate and the gearbox of the knee motor. The plate has mounting holes for the knee limit switches, the optical encoder occlusion plate, and cables as well as holes for mounting to the knee motor.

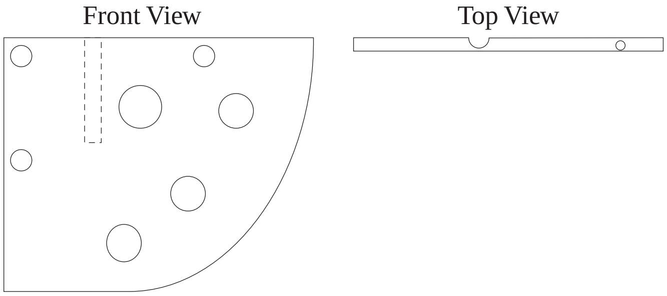


Figure 6: Knee Mounting Plate

Using the new design, a prototype was once again built and tested. When it was deemed acceptable, six more copies were made. Extreme effort was made to ensure that all copies were identical. The prototype was kept to be used for testing purposes. The following is an in-depth analysis of the leg design from the bottom up.

The foot switch is a Hall-effect microswitch activated by a lever which depresses a button on the switch body. This is mounted around the leg with two 0.875” 4-40 screws through 0.125” holes to a 0.75” x 1.25” x 0.125” plate. The foot-switch mounting plate is cleared near its center for a 0.25” 4-40 machine screw which attaches it to the lower leg. The side of the foot-switch mounting plate which contacts the leg has a groove milled on the long dimension to cradle the 0.25” diameter shaft.

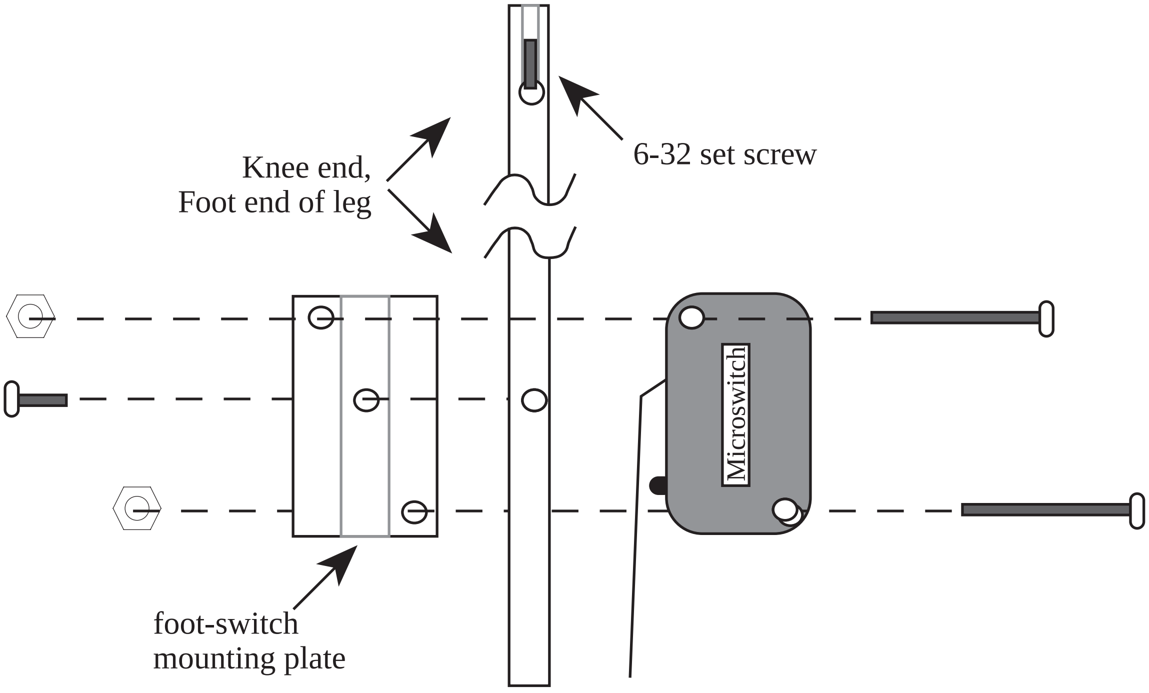


Figure 7: Foot Switch

The leg itself is an 0.25” diameter aluminum rod approximately 5.0” long. The tapped hole for the foot-switch mounting plate is about 1.75” from the leg’s lower end. The 0.125” hole for the shaft of the knee motor is 2.75” up from there (4.5” from the floor if the leg is vertical). The leg is fixed to the flatted shaft of the knee motor with a 0.25” 6-32 set screw through a tapped hole in the top end of the leg.

The knee-hip junction is accomplished with a flat 1” x 1.5” x 0.125” aluminum plate with holes positioned and cleared for mounting to the face of the knee gearbox. The plate has a 0.125” groove in the side of the plate which is mounted against the knee’s gearbox. The shaft of the hip motor can then be pinched between the plate and the gearbox. The knee-hip plate also has three 0.25” holes within an inch of the knee shaft. Two of these (one beneath and one to the right of the knee shaft) hold upper / lower limit switches and the third is cleared for foot-switch wiring. Then one last hole must be added to the knee-hip plate; a hole to accommodate 0.25” 4-40 machine screw must be drilled and tapped in the top edge 1.0” from the groove for the hip shaft. This last hole is for securing the hip’s occlusion plate. The hip motor is simply mounted by the face of its gearbox to the chassis, with its shaft protruding below the body.

The most difficult portion of the leg assembly is the hip position feedback implementation. Convenient material for the optical encoder disk is 0.040” sheet aluminum. It is important that this disk have a constant outer radius of 1.25”. A 135° arc of the circle must be cut to a smaller radius. The edges of this arc trigger the hip limit switch at either end of its range of motion. Opposite the smaller radius arc, the disk is slotted with a bandsaw at intervals of about 17° to put 8 slots in a 135° arc. The hole at the encoder center is cleared to 0.125” for the hip shaft. 1.0” from this, on the line including the center and parallel to that line defined by the limit-switch trigger corners, a hole is cleared for the 4-40 screw which fixes the hip encoder disk to the knee-hip plate.

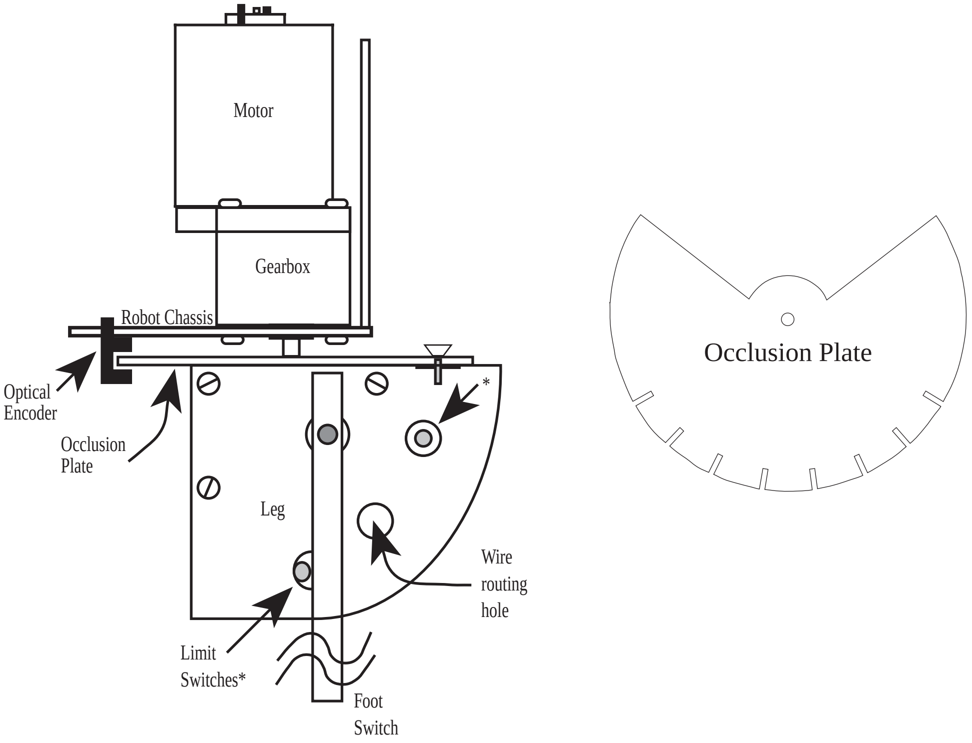


Figure 8: Final Leg Design

Once the leg design was well under way, design of the chassis began. Several considerations were involved in the layout of the chassis. First, it was necessary to determine what sensors would be used to ensure that they would have good clearance. Furthermore, it was necessary to estimate the total size of the electronics that would be mounted on the inside of the robot. Other considerations such as battery size, leg location, and balance also impacted the design. The front of the chassis was given a curvature to provide a good spread for the infrared emitters. The sides of the chassis were designed to be as high as the motors and allow for a cover. Also, access to the control processor was provided through a data port at the rear of the chassis.

The chassis was cut and formed from 0.040" thick sheet aluminum. Joints are secured with short 4-40 machine bolts and nuts. The resulting shape is a rectangle with 45° angled rear corners and a 60° arc of a circle at the front. For simplicity, all legs and mount regions on the chassis are identical; the right side of the robot is*not* a mirror image of the left. Each leg mount region has the basic holes for mounting the hip motor. From a top side viewpoint, there is a 0.25" diameter hole for the limit switch centered 1.125" right of the hip shaft hole. Between these holes is another 0.25" hole for wiring to the knee and foot. A rectangular hole for an "ear" of the optical sensor is 1.25" left of the center of the hip shaft hole; a nearby hole accommodates the wires for this sensor. When the ear of the optical sensor is pushed through the rectangular hole and the housing is flush against the underside of the chassis, the mounted occlusion plate passes smoothly between the sensor emitter and receiver as the hip shaft turns. This first design proved adequate, and it was quickly machined and bent into shape. (see **Appendix C** for a diagram)

With the chassis complete, design moved to sensory input. The chassis design specified where most sensors would be located. The more complex design considerations included sensor range, sensor validation, and power consumption. The design and behavior goals guided the sensor choices. Initially, six sensors were chosen: Infrared proximity sensor, microphone, photocell, micro-switch, sonar, and Pyroelectric sensor. The infrared proximity sensor consists of an IR emitter and receiver pair. The emitter is simply a 940 nm IR LED which is pulsed at 40 KHz. The receiver is tuned via a PLL to detect 940 nm light at 40 KHz; the Sharp GP1U52X receiver was chosen because it included the IR photodetector, PLL, and an amplifier in a single package. The chassis design allowed six detectors to be mounted on the front of the robot and two on either side. The redundant sensors help the robot be fault tolerant since only a small percentage of emitted light will be reflected. To increase the emitted signal strength and further improve fault tolerance, two emitters are located on either side of a receiver. The IR sensors allow Behavior 1 to be easily and reliably implemented. The IR emitter LEDs require approximately 50 mA each, therefore an interface was designed to allow selective control of which emitters are on at a given time. The 555 timer was chosen to generate the 40 KHz pulses needed to drive the LEDs. The microphone allows the robot to monitor and locate sounds in its environment, this allows Behavior 4 to be easily implemented. Two unidirectional microphones, to be mounted approximately six inches apart, were chosen to allow the robot to calculate sound direction via differential sound intensity. Each microphone is amplified with a LM386 op-amp which is fed to the A/D ports of the microprocessor. Photocells allow the robot to detect light; two CdS cells were chosen to allow differential calculation of light intensity thereby allowing Behavior 3 to be implemented.

The remaining three sensors were not implemented due to cost and time considerations; however, a brief description of their purpose is included. In order to improve fault tolerance in Behavior 1, micro-switches were included in the initial design. Two switches would have been mounted on the left and right front plate of the chassis; a "whisker" attached to the switches signals immanent collisions. The whiskers allow detection of objects that do not reflect IR light very well. To even further improve Behavior 1 and to aid in Behavior 5, a sonar sensor mounted on a stepper motor was also included in the original design. A Polaroid Sonar Ranging System was chosen since it can interface with the microprocessor with only one transistor and two inverters and was readily available. The system allows distance determination from 10.5 inches to over 35 feet with accuracy to within one half inch. However, the system does require a special power system, and considerable processor time. Finally, a Pyroelectric sensor was desired to allow the robot to identify human heat patterns which would allow Behavior 5 to be elegantly implemented. Pyroelectric sensors are most commonly used in IR alarm systems since they can detect heat sources in the human IR range; furthermore, the sensor can detect motion and the direction of motion. The Pyroelectric sensor interfaces easily with the microprocessor, but has a prohibitive cost.

The final sensor arrangement consisted of 10 IR receivers (six in front, and two on both the left and right sides), 15 IR emitters (seven in front, and four on both the left and right sides), two microphones (mounted directly to the front of the robot, each with a semi-parabolic cover), and two photocells (mounted horizontally next to the microphones). These sensors provide a good compromise between sensor information, accuracy, cost, and development time.

As mechanical hardware design progressed, choices needed to be made concerning the microprocessor and support hardware. Before the leg design was complete, a decision was made to use the Motorola 6811 microprocessor to control the legs. This decision was based on a cost versus performance analysis; the 6811 microprocessor includes built-in hardware to handle A/D conversions, multiple timed interrupts, timed events, and interprocessor communication as well as being low power and easy to interface. The M68HC811E2 series microprocessor was chosen since it contains 256 bytes on on-chip ram and 2048 bytes EEPROM which allows the microprocessor to operate alone without support / interface chips. The initial specification was for one chip to control all six legs; however, it quickly became apparent that each leg could require a large percentage of the processor's time. Furthermore, the number of external pins on each microprocessor was limited. Therefore, it was decided that each pair of legs (left / right) would be controlled by a separate 6811 microprocessor; this decision allows considerable flexibility with a moderate cost increase.

A separate 6811 microprocessor was also chosen to control the sensors. This choice allows for sensors to be easily extendible (when the initial choice was made, the time intensive sonar sensor was still being considered). The sensor processor is responsible for detecting immanent collisions and notifying the control processor as well as performing differential sound and light intensity calculations.

For the control processor, a Digital Signal Processor (DSP) was initially considered, but was eventually rejected due to the complexity of interfacing the processor with the other microprocessors and the complexity in effectively programming the device. Another M68HC811E2 microprocessor was chosen. The control processor is configured in an expanded mode which allows up to 64K of external memory to be used allowing for complex control code. All microprocessors are clocked with separate 8 MHz clocks, with a 2 MHz bus for memory access. In addition, the on-chip Serial Peripheral Interface (SPI) system can be used to allow serial message passing between processors.

For each 6811 microprocessor, a printed circuit board containing all electronics necessary to run the processor was purchased. These boards contain the additional logic for accessing additional "off-chip" memory, RS232 communication, resetting the microprocessor, and handling control signals. Each board vastly simplifies integrating the 6811 into the robot.

In order to simplify motor control and interfacing, L293B four channel push-pull drivers were chosen to control the motors. Each integrated circuit can be configured as two full H-bridges with a minimal addition of eight diodes per chip. The chips allow direct interfacing with the 6811 microprocessor. Each H-bridge has a motor enable and a direction control, as well as a separate power supply (up to 2A per channel) for driving the motors. Therefore, one chip can control a complete leg, allowing independent hip and knee motions.

The control processor also requires several chips for memory interface (which were included in the 6811 printed circuit board): 74HC573 D-Type Latch and 74HC245 Bus Transceiver used to demultiplex the low address bits and data and the 74HC541 Buffer used to buffer the high address bits. These chips control the address and data fed to the 62256-10 32K RAM and 27C256-15 32K EPROM. In addition, a MAX232 chip is used to allow the control processor to communicate with a remote computer via a RS232 connection.

Two types of Programmable Array Logic (PAL) chips were used to provide custom functions. These chips handle interrupt decoding, memory decoding, SPI communication decoding, and other simple decoding functions. The PAL16L8 provides eight combinatorial logic outputs, while the PAL16R4 has four such outputs and four flip-flop outputs.

Software:

Design for the software used to control the leg hardware began while the legs were still under construction. To simplify the testing process, a prototype leg was built and interfaced with a Motorola M68HC11EVB Evaluation Board. The board and its support software allowed easy simulations of the developing software control systems. To meet the previously specified goals, assembly language was chosen to implement the leg software. Each leg microprocessor is setup in single-chip mode, which allows the code size to be a maximum of 2048 bytes. The on-chip RAM is only 256 bytes and must satisfy the needs for both variables and stack space; therefore, careful design is needed to avoid stack overflow and unpredictable results.

Careful consideration of the hardware available from the 6811 microprocessor led to the choice of an interrupt driven system. The M68HC811E2 microprocessor has six general external interrupts: !IRQ, !XIRQ, and four Input Capture (IC) pins which may be configured as interrupts. Each leg microprocessor must control a left / right pair of legs which means each leg can have a maximum of 2 identical interrupts. The !XIRQ line is used to implement fault-tolerant system recovery and the !IRQ line is unused. Using the IC inputs as interrupts, each leg has an interrupt which indicates that a movement limit (right / left, up / down, or floor) has been reached and an interrupt which indicates that the optical sensor used to track hip movements has triggered. In order to differentiate between the three movement limits, each limit signal is also sent to an input pin on the microprocessor. When the interrupt is triggered, the interrupt handler reads the pins to determine which limit(s) triggered the interrupt as shown below:

Port B	Bit	7	6	5	4	3	2	1	0
	Control Function	Right Hip Direction	Right Hip Enable	Right Knee Direction	Right Knee Enable	Left Hip Direction	Left Hip Enable	Left Knee Direction	Left Knee Enable
Port C	Bit	7	6	5	4	3	2	1	0
	Active Interrupt	-	Right Hip	Right Floor	Right Knee	-	Left Hip	Left Floor	Left Knee

Figure 9: Interrupt De-Multiplexing

The code must be carefully designed to allow proper handling of coincident triggers. In order to facilitate quick handling of interrupt routines, the leg code uses the wait (WAI) instruction to wait for an interrupt when it has nothing else to do; the wait instruction prepares for an interrupt by pre-storing all data so that there is virtually no delay in servicing an interrupt.

The timer overflow (TOF) interrupt, an internal interrupt, is used to tell the microprocessor that a specified period of time has elapsed. Using the TOF interrupt and four Output Compare (OC) pins, it is possible to easily implement pulse width modulation (PWM). Using PWM allows MuLIR to conserve power while driving the legs in an efficient manner. The OC pins are similar to the 6811's IC pins. These pins allow for the output state of a pin to be changed when an internal state meets a pre-defined trigger state and to optionally generate an internal interrupt. To implement PWM, the leg code sets the TOF interrupt to trigger with a period of 32.768 ms or approximately 30.5 times/second. When an overflow occurs, the 6811 generates an interrupt which turns on the motor output and resets the OC to occur at some percentage of the 32.768 ms period (theoretically this percentage is represented by a 16-bit value between [1, 65535], but interrupt overhead and minimal required motor torque necessitate a minimum of 19000). When the OC state matches the trigger state, motor output is halted. This implementation defines Pulse-Width Ratio from 30% to 100%.

The optical hip sensor produces interrupts whenever the hip rotates through a slot in the occlusion plate. The plate was designed to contain 2^n evenly spaced slots to allow simple error checking (in the current design n = 3). However, the use of the optical sensor does present several problems which have remained unsolved in this implementation. The main problem is that the system cannot tell the difference between clockwise and counter clockwise rotation. The motor's rotation is controlled by an output from the microprocessor. By controlling the motor's direction of rotation, the microprocessor assumes that the interrupts from the hip sensor are occurring in the same direction; however, if for some reason the hip rotates in the opposite direction (due to slippage or weight when the hip is not moving) then the system still assumes that the rotation is in the original direction. Furthermore, if the hip stops while centered on a slot in the occlusion plate, then minute fluctuations can create false interrupts. Due to these limitations, the optical hip sensor is not relied on by the main control system. Hopefully, future implementations will be able to correct these problems.

Once the leg microprocessor code had been developed and tested, design moved to the sensor microprocessor. The four IC interrupts were used to notify the processor when the IR detectors indicate a nearby object. One interrupt connects to the IR detectors on the left side of the robot, one to the detectors on the right side, one to half the detectors on the front right side, and the other to the detectors on the front left side. When an object is detected, interrupt code is notified; with this configuration there is no need to continually poll the IR detectors to implement object avoidance. The microphones and photocells are polled at constant intervals. The code reads a value from the left microphone or photocell and then from the right. The difference between the two values is used to indicate the direction of the sound / light source. A running average of the sound / light levels is maintained to allow background noise to be eliminated. When the input is above a variable threshold value (and lasts longer than a minimum delay), the sensor microprocessor notifies the control microprocessor that an appropriate action needs to be taken. In order to conserve power, the sensor microprocessor also enables the IR emitters only when avoidance is needed (for example, the left / right side emitters are only needed before MuLIR attempts to turn left or right). The code for polling the microphones and photocells was optimized to be as fast as possible to allow for a high sampling frequency (≈ 16 KHz, which by the Nyquist Theorem means frequencies up to about 8 KHz may be sampled).

Due to the code size restrictions on the leg and sensor microprocessors, the control microprocessor must make all decisions. The leg and sensor microprocessors pass information to the control microprocessor via the 6811 SPI system. The control microprocessor then sends commands to some subset of the microprocessors. The SPI system is a serial connection shared by all processors. In order to send information, a processor must first become master while the other processors become slaves. The five microprocessors are thus linked in a multiple master - multiple slave system Conflicting masters are prevented via hardware support in the PAL16L8. To become master a processor checks to see if any other processor is current master. If one is, then the master to be enters a spin lock until mastery is free. Once mastery is free, the microprocessor asserts the master pin which tells the remaining processors that it is master. When the master sends data the slaves queue off its clock (allowing data transfer approaching 2.0 Mbits / sec). When data is received an SPI interrupt is generated. Within the interrupt routine, the receiving processors check that the data was addressed to them. If not, it is ignored, otherwise the data is stored in a command queue to be acted upon on the next command check. Using an interrupt driven system insures that a processor cannot miss commands. When the command is processed, any additional information needed is read. This is actually a race condition and could result in problems if multiple interrupts occur before the receiving processor can read the data. The worst case of this race condition occurs when the control processor tells a leg processor to notify it when a limit interrupt is generated. If the leg processor receives a new command and then generates the limit interrupt (which disables the SPI interrupt) before it can read in the data for the new command; the leg processor waits to become master while the control processor is sending the data. This additional data is lost; once the leg processor sends its notification to the control processor, it waits for the additional data which has been lost resulting in a deadlock. The probability of such a situation is low, and the control processor can reset the deadlocked processor, but a better SPI algorithm would be a more practical solution.

The control processor's code was written in C to facilitate code readability and modification. The code implements a subsumption architecture which requests information from the other processors and activates behaviors based on this information. The control processor makes all motion decisions and is responsible for implementing coordinated movement using hardcoded walking patterns. The control processor sends messages to other processors via SPI and optionally to a remote computer over a RS-232 link via the Serial Communications Interface (SCI). The SCI system allows a user to monitor behaviors implemented by the control processor, get register and memory dumps from any processor, as well as download new code to replace the operating code of any processor. The SCI interface allows quick debugging and code replacement. The control processor supports the following SCI commands:

'0'	Select Main Processor
'1'	Select Front Leg Processor
'2'	Select Middle Leg Processor
'3'	Select Back Leg Processor
'4'	Select Sensor Processor
'5'	All Legs

'A'		Alive (Ping Processor)
'D'	<Start Address> <# bytes>	Download
'E'	<Start Address>	Execute
'H'		Halt Motion
'L'	<L/R/B> <0-8> <0/1>	Move Leg
'M'	<Start Address> <# bytes>	Dump Memory
'P'	<L/R/B>	Retrieve Leg Position
'R'		Dump Registers
'S'		Retrieve Sensor Information
'?'		Command List

The user must first choose a processor, and then may use the commands to change the state of that processor. In order to prevent user mistakes, the code in the microprocessor which implements code replacement cannot be replaced. This ensures that a programmer mistake cannot permanently disable the system. Furthermore, if the control processor can determine that another processor is not responding, it can reset that processor via the !XIRQ interrupt line.

While much care was taken in the design of the software control system, several problems exist which should be summarized again. First, it is possible for a processor to become deadlocked due to a race condition in the message passing code. Second, the optical hip sensor may not be reliable due to the discrete nature of the signal. Replacing this sensor with a continuous device such as a potentiometer would solve this problem. Last, it is possible for the robot's IR detectors to miss objects causing a collision. This is due to the fact that not all objects reflect infrared light well; adding "whiskers" to the front of the robot would most likely result in a more perfect solution. It is important to stress the necessity for fault-tolerant systems in the design of arobotic system. Many malfunctions can occur in a system that is subjected to unknown conditions. While care was taken to implement a fault-tolerant system, improvements are always possible.

Theory

Hardware:

The M68HC811E2 microprocessor consists of 11 distinct parts: Mode Control, Clock Logic, Timer System, Bus Expansion / Parallel I/O Multiplexer, Serial Peripheral Interface, Serial Communication Interface, EEPROM, RAM, and the CPU. The 6811 is available in a number of different packages; this project used the 52-pin PLCC. (See **Appendix B** for a pinout diagram) The 6811 has 110 instructions (requiring from 2 to 41 cycles), 57 registers, one 16-bit timer, 21 interrupt sources, an eight channel, 8-bit A/D converter, up to 62 bits of I/O (configuration dependent), and supports six addressing modes (Immediate, Direct, Extended, Indexed, Inherent, and Relative). Of the 57 registers, two are general-purpose, two are 16-bit index registers, three more implement the stack pointer, program counter, and condition code registers while the rest are used as output, control, and status registers. The basic programming model follows:[2]

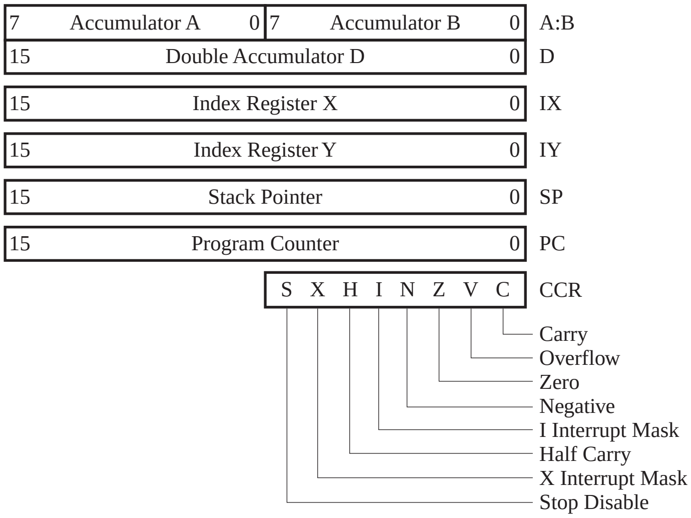


Figure 10: 6811 Programming Model

Accumulators A and B are general-purpose 8-bit registers (they can also be treated as a single 16-bit register, D). Index Registers X and Y provide a 16-bit indexing value that can be added to an 9-bit offset provided in an instruction to create an effective address. They may also be used as counters or temporary storage registers.

The 6811 is capable of operating in four modes: Single Chip, Expanded, Bootstrap, and Special Test. Bootstrap mode is used to download initial startup code to a chip. Special Test mode was not used in this project. The leg and sensor processors run

[2] HC11: M68HC11 E Series. Motorola. p. 3-2.

in Single Chip mode while the control processor runs in Expanded mode. The memory map for the three modes is shown below.[3]

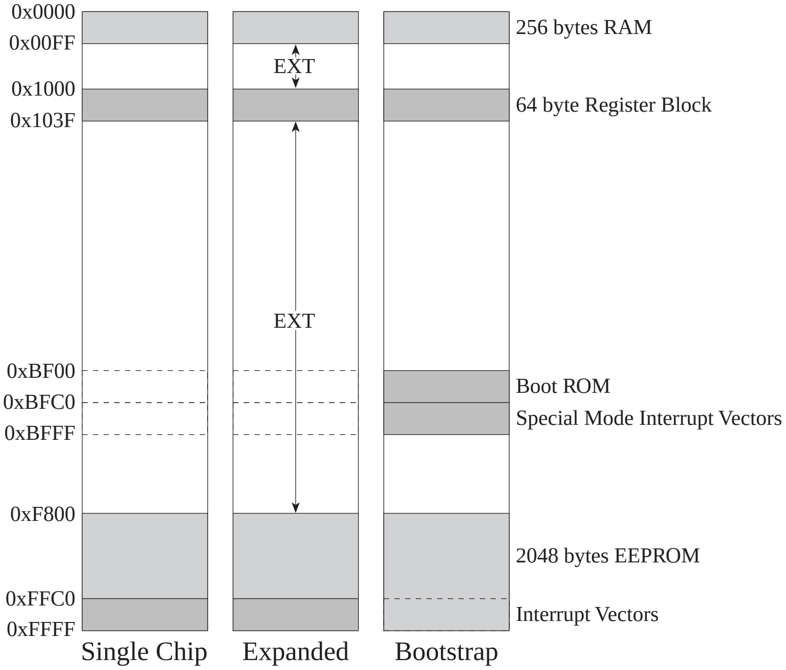
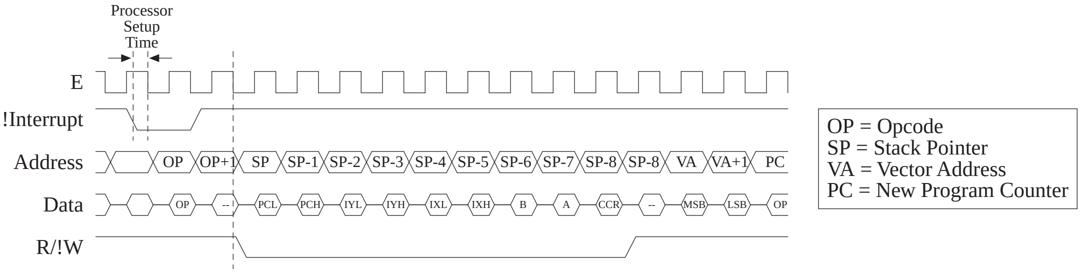


Figure 11: 6811 Memory Map

After a reset (either External through the !RESET pin or a Power-On Reset), all system values are set to a default state and the system jumps to the address stored in the reset vector located either the Interrupt Vector Table (Single Chip or Expanded mode) or the Special Mode Interrupt Vector Table (Bootstrap or Special Test mode). The address for all interrupt routines are stored in these tables. When an interrupt occurs it is serviced following a hardware priority which ensures correct handling of multiple interrupts. Six of the 21 interrupts are non-maskable and are handled in the following order: !RESET, Clock Monitor reset, COP watchdog reset, !XIRQ, Illegal opcode interrupt, Software Interrupt (SWI). The remain 15 interrupts may be selectively enabled via software, and are handled in the following order: !IRQ, Real-time interrupt (RTI), IC1 - 3, OC1-4, IC4, Timer Overflow (TOF), Pulse Accumulator overflow, Pulse Accumulator input, SPI, SCI. Any one of these interrupts may be given the highest priority; however, for MuLIR, the default priority is optimal (changing the priority will cause serious problems). Interrupt timing and ordering of all CPU registers on the stack are shown in the following diagram:[4]

[3] ibid. p. 4-5.

[4] ibid. pp. A-10, 5-12.



Memory Location	CPU Registers
SP	PCL
SP-1	PCH
SP-2	IYL
SP-3	IYH
SP-4	IXL
SP-5	IXH
SP-6	ACCA
SP-7	ACCB
SP-8	CCR

Figure 12: Interrupt Timing and Register to Stack Allocation

When a user requests a register dump, the main processor has the indicated processor send its registers in this same order. Furthermore, a pre-emptive multitasking system can be easily implemented on the 6811. The CPU's state can then be stored simply by causing an interrupt, choosing a new stack, and returning. The Interactive C (**IC**) compiler uses this method (available via anonymous ftp from the MIT Media Laboratory server cherupakha.media.mit.edu). Future projects would benefit from using a multitasking subsumption architecture.

In order to speed interrupt processing, leg and sensor code pre-stores the CPU registers on the stack using the wait (WAI) instruction. The wait instruction pushes the registers on the stack but does not return until it receives an interrupt. When the interrupt occurs, it may be immediately serviced by finding the Vector Address and jumping to it. Use of the wait instruction cuts the delay time to service an interrupt from 15 cycles to a typical value of 3 cycles.

Serial communication with an external computer is provided through a universal asynchronous receiver / transmitter (UART) on board the 6811. The UART sends control signals to the MAX232 which turns the signals into RS232 signals. The control processor may then be communicated with directly using ASCII control codes. The timing diagram for the SCI interface is presented below (*NOTE*: This is not an RS232 timing diagram).[5]

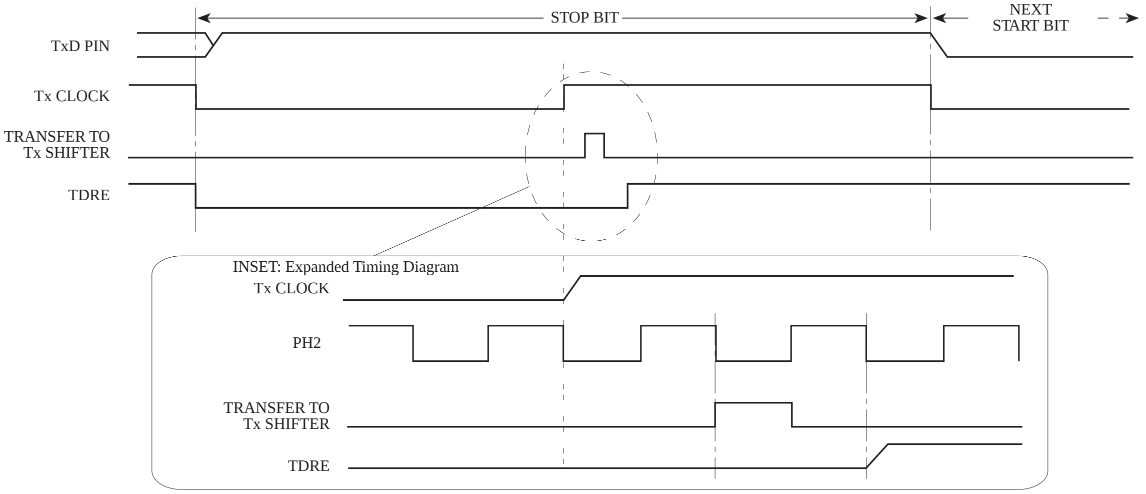


Figure 13: Serial Communication Interface (SCI) Timing

Interprocessor communications are implemented using the SPI system. The SPI commands are a subset of the SCI commands. Processors communicate through a one-way serial stream (the connection is actually full duplex, but in this implementation, only the master processor sends data). In order to send data, a processor must first become a "Master." The process of becoming a master is handled in hardware with a PAL16L8 To become a master, a processor checks if anyone else is master by reading the value on the !SS (Slave Select) pin. If !SS is high, then the processor is free to become master (ie. it is not currently a slave). The processor then asserts S, its processor select pin (pin PD0 for the leg and sensor processors, pin PA4 for the control processor), which goes to the PAL16L8. The PAL then performs these combinatorial logic operations to select the other processors:

$$
\begin{aligned}
!SS_{\text{control}} &= S_{\text{front legs}} + S_{\text{middle legs}} + S_{\text{back legs}} + S_{\text{sensor}} \\
!SS_{\text{front legs}} &= S_{\text{control}} + S_{\text{middle legs}} + S_{\text{back legs}} + S_{\text{sensor}} \\
!SS_{\text{middle legs}} &= S_{\text{control}} + S_{\text{front legs}} + S_{\text{back legs}} + S_{\text{sensor}} \\
!SS_{\text{back legs}} &= S_{\text{control}} + S_{\text{front legs}} + S_{\text{middle legs}} + S_{\text{sensor}} \\
!SS_{\text{sensor}} &= S_{\text{control}} + S_{\text{front legs}} + S_{\text{middle legs}} + S_{\text{back legs}}
\end{aligned}
$$

When a processor is a slave, its !SS pin goes low; if the master's !SS pin goes low, it is erroneously being selected as a slave (this can only happen if two processors request to be master simultaneously). In this case, both processors SPI systems are reset and

[5] HC11: M68HC11 Reference Manual. Motorola. p. 9-33.

neither becomes master (this is done to prevent Latch-Up which can occur if both processors try to drive the SPI system). In order to proceed they must both reset their SPI systems and try again. To further reduce the chance of damage due to multiple processors becoming masters, the SPI system is configured in Wired-OR mode. In Wired-OR mode, no active high signals are produced; 10K resistors between all SPI and SCI lines and power produces high signals when the line is not actively being pulled low. The timing diagram for the SPI system is shown below:[6]

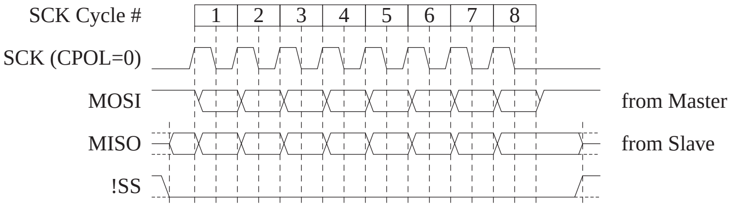


Figure 14: Serial Peripheral Interface (SPI) Timing

The 6811's timing system is used to implement Pulse Width Modulation for leg motor control. The 6811 contains a 16-bit timer which can generate an interrupt when it overflows (when the timer goes from 65535 back to 0). When running with an 8 MHz clock, the timer overflows every 32.768 ms. When the timer overflows, the interrupt routine turns all active motors on while setting the appropriate OC values to turn the motors off when the system clock reaches the value stored in the OC register. To generate enough torque to trigger the limit switches, these values must be greater than or equal to 19000. The following is a diagram of the PWM waveform (it should be noted that the frequency never changes, just the on / off times which define the duty cycle):[7]

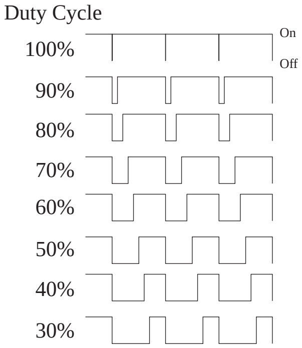


Figure 15: Pulse Width Modulation (PWM) waveform

L293B integrated circuits are used to implement H-bridges. Each circuit can be used to implement two full H-bridges. The enable pin of each H-bridge is the logic AND of the Motor Enable from Port B and the PWM enable from Port A. This implementation

[6] HC11: M68HC11 E Series. Motorola. p. 8-3.

[7] McComb. p. 104.

simplifies motor control by allowing the PWM timing software to be running continuously. A separate subroutine is used to set the Port B Motor Enable pin high or low.

The Input Capture and Output Compare pins used to register leg interrupts and imminent collisions and create the PWM waveforms have the following timing.[8]

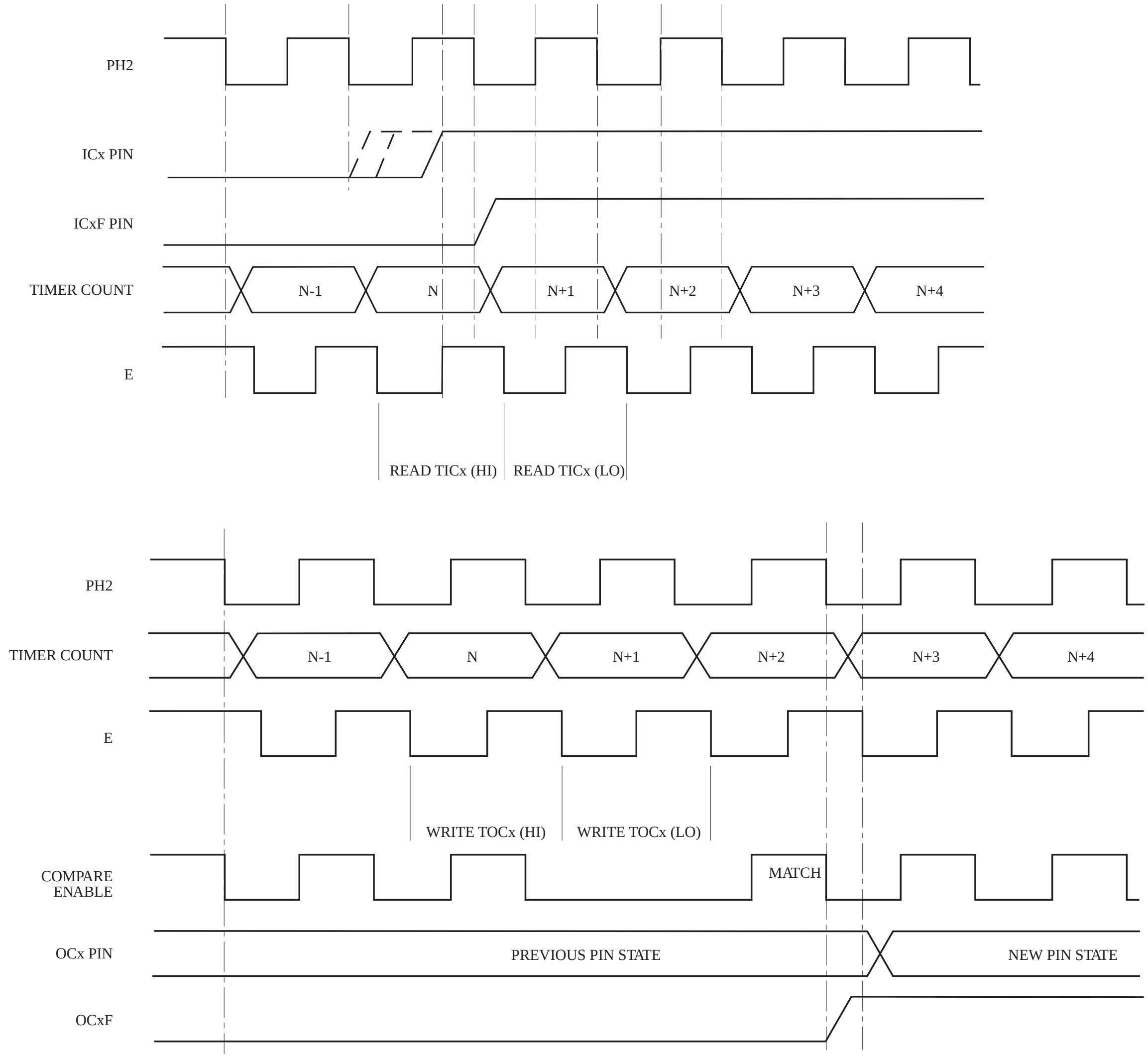


Figure 16: Input Capture (IC) and Output Compare (OC) Timing

The Port B pins used to implement Motor Enable and Direction Control are output only. Reading Port B will return an 8-bit value representing the state of the pin, but this is an internal value. Port C is an input/output port; it is used to determine which

[8] HC11: M68HC11 Reference Manual. Motorola. pp. 10-39, 10-40.

interrupt(s) occurred when an IC interrupt (Port A) occurs. (See Also: Figure 9) The timing for reads / writes is shown below:[9]

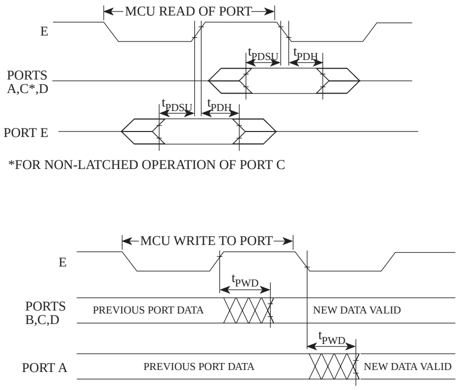


Figure 17: Port Read / Write Timing

The timing for port reads / writes and interrupts is extremely critical, any changes to the interrupt system must ensure that race conditions do not occur. With the use of the wait instruction, the delay for processing an interrupt is reduced to approximately three cycles, while it takes a minimum of four cycles (maximum of eight) for data to become steady for a port read. This means that interrupt routines that read common signals (the same signal that caused the interrupt is also being read) should wait a minimum of three cycles before reading the value to avoid any chance of reading a bad value.

The Analog-to-Digital system (8-channel, 8-bit) is built into Port E of the 6811. The system is capable of reading from up to eight different sources (one at a time). The A/D system includes a sample-and-hold circuit to reduce the required external circuitry. Each conversion is a sequence of eight comparison operations, beginning with the most significant bit (MSB). A successive approximation register (SAR) is used to store the result of each successive comparison. When a conversion is complete, the contents of SAR are transferred to the appropriate result register. There are four result registers, and the 6811 can be initialized to perform up to four consecutive conversions. When these are complete, the conversion complete flag (CCF) is set. The timing for the A/D conversion sequence is shown below:[10]

[9] HC11: M68HC11 E Series. Motorola. pp. A-12, A-13.

[10] ibid. p. 10-5.

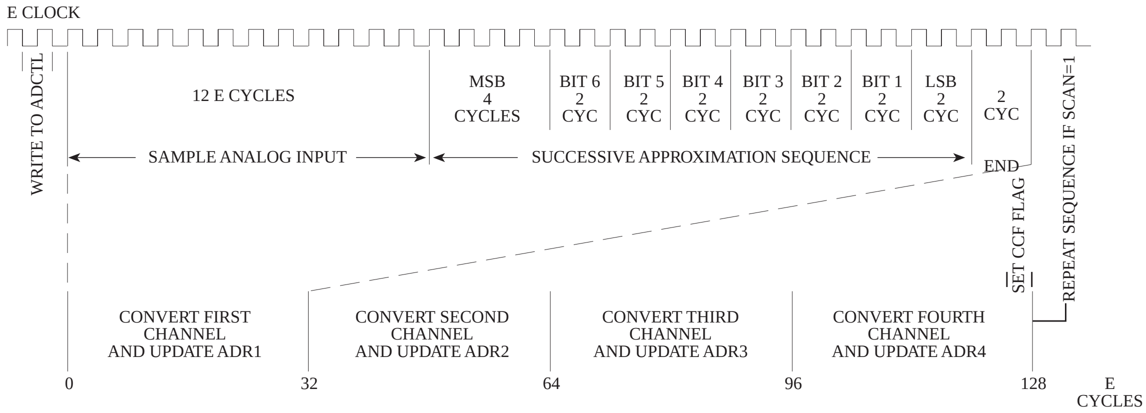

E CLOCK
WRITE TO ADCTL
12 E CYCLES
MSB
4
CYCLES
BIT 6
2
CYC
BIT 5
2
CYC
BIT 4
2
CYC
BIT 3
2
CYC
BIT 2
2
CYC
BIT 1
2
CYC
LSB
2
CYC
2
CYC
SAMPLE ANALOG INPUT
SUCCESSIVE APPROXIMATION SEQUENCE
END
SET CCF FLAG
REPEAT SEQUENCE IF SCAN=1
CONVERT FIRST CHANNEL AND UPDATE ADR1
CONVERT SECOND CHANNEL AND UPDATE ADR2
CONVERT THIRD CHANNEL AND UPDATE ADR3
CONVERT FOURTH CHANNEL AND UPDATE ADR4
0
32
64
96
128
E CYCLES

Figure 18: Analog-to-Digital (A/D) Timing

The A/D system is used for monitoring sound and light intensities. Two microphones and two photocells are monitored continuously. The A/D is initialized to perform four consecutive reads: left microphone, right microphone, left photocell, and right photocell. Once the values have been read, the CCF flag is set, and the system may begin processing the results. At any time during the conversion, the IR system can generate an interrupt, if this happens a variable is set which discards the intensity values and the control processor is notified of the interrupt.

The control processor is the only system that uses the 6811's expanded mode In expanded mode, up to 64K of external memory may be accessed. The memory is overlaid by the internal registers, RAM, and EEPROM as shown earlier. The internal RAM is left active since it has a faster access time and may be used for important variables. The lower 32K of external memory is filled with a 62256-10 static RAM module. This allows for extensive variables or possibly for an entire new control system to be downloaded. The upper 32K of external memory is filled with a 27C256-15 EPROM module seated in a TEXTOOL Zero Insertion Force IC socket for easy access. The EPROM is used to hold permanent copies of the control code. The timing for accessing both memory modules is shown below:[11]

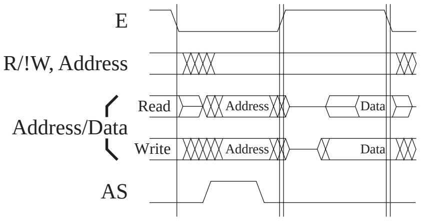


Figure 19: Expanded Mode Read / Write Timing

[11] ibid. p. A-20.

In addition to the previously mentioned integrated circuits, six PAL16R4 circuits are used to provide interrupt signals to the leg processors as well as some other functions The PALs are needed since the interrupt generated by a limit switch must last at most for one E clock cycle. This allows for all limit switches to trigger interrupts while others are still being held down. The logic for the PAL is shown below:

```
UD_Opp_Dir_Out = !UD_Opp_Dir_In
FB_Opp_Dir_Out = !FB_Opp_Dir_In

UD_int.D       = Up_Dn_Int
Floor_int.D    = Floor_Int
FB_int.D       = Fwd_Bkwd_Int
Intr.D         = (!UD_int & Up_Dn_Int) + (!Floor_int & Floor_Int) +
                 (!FB_int & Fwd_Bkwd_Int)
```

The first two values are used by the direction inputs of the L386B integrate circuit to create a H-bridge. The next three values are inputs to D flip-flops from the three limit switches (Up / Down, Floor, and Forward / Backward). The last entry is the generated interrupt signal which is high for only one E clock.

Each H-bridge controls a single motor. The right and left sides of the robot have separate power supplies. In order to determine the battery characteristics required for worst case activity, the 12 motors used in the legs were measured for running current and stall current for three voltages. The average and maximum values were then plotted to determine the motor characteristics. (See Figure 20 next page) It is theoretically possible for all motors on one half of the robot to be operating at the same time. Therefore, the battery must provide from 720 mA (at 5V) to 960 mA (at 15V) maximum; furthermore, if a motor becomes stalled, it will approximately double its current draw. Even though the motor will only be stalled momentarily, the battery should be able to handle the increased current draw due to stalling. A battery with 1440 mA to 1920 mA per hour would be ideal. Since the robot has a steady current drain (it is almost continually moving), rechargeable batteries are a necessity. Gelled electrolyte (Gel-Cell) batteries are the best, offering a nearly constant discharge rate until completely discharged, but are expensive and difficult to find. Nickel-cadmium (NiCad) batteries are the next best thing, offering a constant discharge rate until approximately ¾ way discharged when it becomes linearly decreasing. NiCad batteries are cheap and easy to find. Two 7.5V 1.8A/hr NiCad batteries were purchased for the robot from a hobby store (they are typically used for Remote Control cars). A Quick charger was also purchased with the batteries to allow them to be recharged within 20 minutes (the recharge sequence for the Panda Quick Charger is as follows: 20 min. quick charge + 20 min. charger cool-down. This is repeated for all batteries). One note about NiCad is that they can develop *memories* if recharged too soon; the batteries should be recharged only after they enter the linearly decreasing discharge state. Furthermore, allowing a NiCad battery to completely discharge may damage it; therefore, the battery should be recharged as soon as it can be determined that it has entered the linear region.

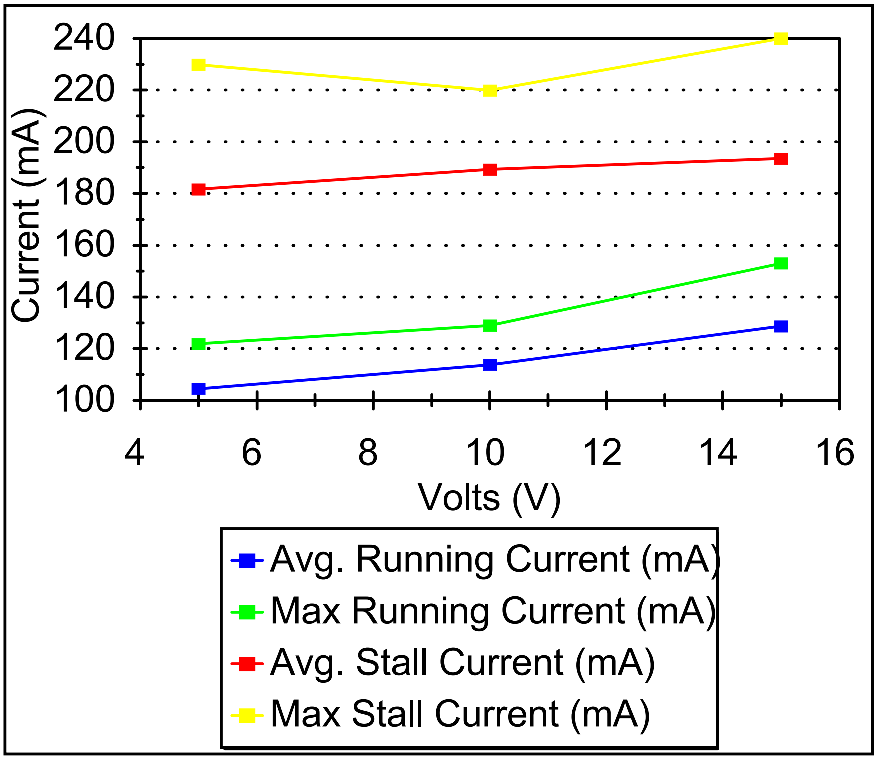


Figure 20: Motor Current Requirements

Biological Basis:

When designing a walking robot and investigating modes of motion, it seems prudent to examine the walkers with the most experience. Biological walkers have spent millennia optimizing their gaits for speed, stability, and adaptability and therefore provide the best guidance for the design process.

Since before 1900, scientists have been studying the locomotion of animals that move with two, four, and six legs. Muybridge, whose studies are classics in the field, analyzed the gaits of horses and humans using successive photographs. Sixty years later, two Yugoslavian scientists used the first mathematical methods for gait description, framing legged motion in terms of finite limb states – planted or raised. Since then formulas, graphic representations, and specific terminologies have evolved to define and further the study of gaits.

The major types of gaits developed through observation and experimentation by investigators are either periodic or non-periodic. Periodic gaits include those that occur naturally and provide optimal speed and stability, as well as a few developed for walking machines. Non-periodic gaits do not appear in nature; these are all special-purpose human inventions created for application in the aforementioned machines. Song and Waldron offer a classification scheme for these gaits from a mechanical rather than biological standpoint. The wave gait shown below corresponds to the metachronal one for insects:[12]

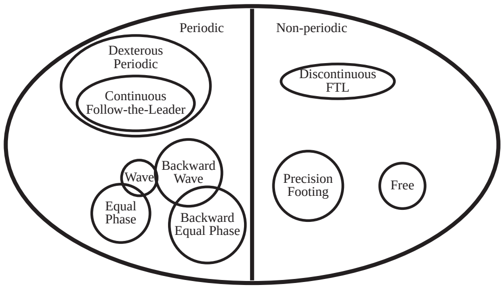


Figure 21: Gait Classification

In the development of this project, emphasis has fallen on the periodic gaits, as these promise to be easier to implement and will remain true to the “insect-emulation” goals of the project. Wilson’s 1966 review of insect walking sheds light on the specific manifestations of and physiological reasons for periodic gaits in the biological world.

In the natural world, periodic forms of locomotion can be further divided into legless and legged categories. The former includes “either undulatory or peristaltic waves which move from anterior to posterior as the animal moves forward; the waves move opposite to the direction of locomotion.”[13] The latter may include undulations of the whole body, but these undulations, as well as the wave of limb activation, move along the body *in the direction of motion* (which he terms *metachronal*). These phenomena are generalized, even to the level of protozoans. Those with simple flagella follow the “legless” model, whereas those with complex flagella act as “legged” creatures. The rest of the animal world is readily classified by these means as well; fish fit the legless mold, tetrapods use metachronal waves of limb movement (at least at low speeds), and arthropods follow a metachronal pattern with opposite appendages paired.

[12] Song. p. 40.

[13] Wilson. p. 115.

Wilson notes, “Insects are no exception, but the smaller number of legs tends to obscure this basic fact. Superficial examination of the gaits of insects reveals a bewildering assortment of different patterns.”[14] The gaits apparently differ among species, may alter for a single insect at different speeds, and change depending upon the number of legs allocated (or available) for walking. The apparent plasticity (*plastizität*, as introduced by Bethe) of insects’ nervous systems to allow the observed degree of adaptation, even compensation for amputations, has in part driven the quest for understanding of the mechanisms at work.

Extensive observation and mathematical analysis / explanation of the gaits of insects preceded true physiological understanding of the mechanisms at work. Out of these initial inquiries emerged an apparent set of rules for the coordination of insect motion:[15]

1. A wave of protractions (forward movements of the legs relative to the body) runs from posterior to anterior (and no leg protracts until the one behind is placed in a supporting position).

2. Contralateral legs of the same segment alternate in phase.

And if (based on study of *Carausius morosus*) the further simplifying assumption is made that protraction time is constant, one can further specify that:

3. Protraction time is constant.

4. Frequency varies – retraction time must decrease as frequency increases.

5. The intervals between steps of the hind leg and middle leg and between the middle leg and the foreleg are constant, while the interval between the foreleg and hind leg steps varies inversely with frequency.

Following these five simple rules allows one to logically reconstruct the patterns observed in insects, watching a high degree of complexity develop from a few simple constraints. In the following patterns, the labels R_x refer to the Right side of the insect and x specifies the leg number; the labels L_x refer to the Left side as shown in the figure:

[14] ibid. p. 116.

[15] ibid. p. 117.

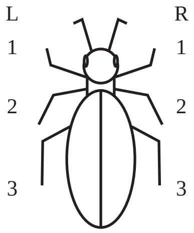


Figure 22: Gait Labels

Rules 1 and 2 give us a basic pattern for the motion of six legs. The slowest version looks like:

I. ...R_3,R_2,R_1.............. R_3,R_2,R_1..............
.................L_3,L_2,L_1.............. L_3,L_2,L_1

– the wave proceeds posterior to anterior and the sides alternate in phase. Speeding this pattern up requires some of the leg actions to overlap temporally, complicating the appearance of the gait.

II. ...R_3,R_2,R_1...... R_3,R_2,R_1...... R_3,R_2,R_1
.... L_3,L_2,L_1...... L_3,L_2,L_1.........

Now the first and last leg actions on alternate sides occur simultaneously. Another speed-up gives:

III. R_3,R_2,R_1, R_3 R_2,R_1, R_3,R_2,R_1,
L_1,L_3,L_2, L_1,L_3,L_2, L_1,L_3,L_2,

Things have gotten very tight here, and a pattern of synchronous diagonal pairs develops. This is significant in that it reproduces, through the simple model, a pattern documented in insects. One more speed-up yields:

IV. R_3)R_2 (R_1R_3) R_2 (R_1R_3) R_2 (R_1R_3)
L_2(L_1L_3) L_2 (L_1L_3) L_2 (L_1L_3) L_2

which is an “alternating tripod” gait. The legs paired in parentheses activate sequentially, but the movements occur so near one another in time that they appear simultaneous. Each front / hind pair on one side activates in apparent synchronization with the middle leg on the opposite side. Wilson notes that “[f]or cockroaches no higher speed gait has ever been reported.”

This would seem to be the limit of this model, but Wendler reports that in the stick insect the delay between the front and hind leg on each side can be*less than zero*, which yields a non-metachronal sequence …R_3 R_1, R_2, R_3 R_1, R_2,…. Wilson points out that this can be counted as an overlap of the basic metachronal sequence:

$$\begin{array}{llllllll}
 & R_3 & R_2 & R_1 & & & & \\
 & & & R_3 & R_2 & R_1 & & \\
 & & & & & R_3 & R_2 & R_1 \\
\hline
\text{V.} & R_3 & R_2 & R_3R_1 & R_2 & R_3R_1 & R_2 & R_1
\end{array}$$

and that these "sequential patterns have been observed in insects. To [his] knowledge, no other patterns, using six legs, have been reported for straight forward walking. A characteristic of the model is that the insect *always stands on at least three legs which enclose the vertical axis through the center of gravity*."[16]

The model outlined above is excellent for developing a basic understanding of insect locomotion and can be applied with minimal error to most adult insects walking with six legs. There are of course modifications which would make it more realistic and beyond those, exceptions and deviations. One is that the interval between any two adjacent legs may also vary with frequency, not just the interval between the fore and hind legs. (Hughes' (1952) more complex model takes this into account.) Another is that insects do not always walk with six legs. Some hold the front or rear legs up and walk with the other four. Other insects use their underside as a point of support. Insects, such as grasshoppers, which have one proportionally large set of legs may step in phase with the odd pair, but at lower frequency.

Additionally, "[t]here is apparent escape from the condition, 'no leg protracts until the one behind is placed in a supporting position,' at the highest running frequencies."[17] Changing conditions such as turning or amputation may also result in deviations from this rule at normal speeds. However, even amputation does not always result in an exceptional gait.

Studies of double-amputee insects show that as a general rule, removal of any two legs (one from each side) causes the insect to adopt the diagonal-pairs gait of a tetrapod. No learning period is required and the phenomenon may appear to be part of some complex walking logic, a rapid adaptation or perhaps plasticity in the nervous system. Closer examination, however, reveals that new gaits may be adaptations of existing gaits.

This is most obvious in the case where the second leg on each side is removed. The resulting gaits observed are easily derived from intact-animal leg sequences I and II

[16] ibid. p. 121. (emphasis added)

[17] ibid. p. 122.

above. In fact, II translates directly to a diagonal-pairs gait (IIa) upon removal of the middle pair of legs.

Ia. ...$\mathbf{R_3}$,$\cancel{R}_2$,$\mathbf{R_1}$.................$\mathbf{R_3}$,$\cancel{R}_2$,$\mathbf{R_1}$........
................$\mathbf{L_3}$,$\cancel{L}_2$,$\mathbf{L_1}$.................$\mathbf{L_3}$,$\cancel{L}_2$

IIa. ...$\mathbf{R_3}$,$\cancel{R}_2$,$\mathbf{R_1}$......$\mathbf{R_3}$,$\cancel{R}_2$,$\mathbf{R_1}$......$\mathbf{R_3}$,$\cancel{R}_2$,$\mathbf{R_1}$
...$\mathbf{L_1}$......$\mathbf{L_3}$,$\cancel{L}_2$,$\mathbf{L_1}$......$\mathbf{L_3}$,$\cancel{L}_2$,$\mathbf{L_1}$......$\mathbf{L_3}$

Close observation of the amputee often reveals that the stumps continue to oscillate in normal phase relative to the other legs. An inefficient form of gait IV can even be elicited by startling insect amputees. The resulting gait moves the intact legs on each side in synchrony, so that the body rocks back and forth, in many cases dragging the underside and hindering progress. The cockroach may execute the only effective post-amputation version of the alternating-tripod scheme (IVa); the pair of legs on each side cycles fast enough to arrest body roll before the abdomen drags.

IVa. $\mathbf{R_3}$), $\cancel{R}_2$,($\mathbf{R_1R_3}$),$\cancel{R}_2$,($\mathbf{R_1R_3}$),$\cancel{R}_2$,($\mathbf{R_1R_3}$)
$\cancel{L}_2$, ($\mathbf{L_1L_3}$),$\cancel{L}_2$,($\mathbf{L_1L_3}$),$\cancel{L}_2$,($\mathbf{L_1L_3}$),$\cancel{L}_2$,

The next step is to determine how these descriptive ‘rules’ are actually manifested in a biological mechanism of coordination. Looking at this model, a simple explanation might be that each leg on a side triggers the next most anterior leg, with the foreleg triggering the hind again. This requires strict adherence to rule 1 above, and does not explain gait V. Restraining one leg of a walking insect quickly eliminates this mechanism as a possibility; the other five legs continue to step in regular patterns, even without the “trigger” presumably required from the still leg. These data suggest that each segment has an oscillatory mechanism whereby sets of legs can function semi-independently of the others.

Experimental biologists have gathered physiological and neurological data that offer better insight into these mechanisms. Legs of an isolated segment can step alternately. This seems to correspond to the anatomical arrangement of the arthropod central nervous system as shown below:

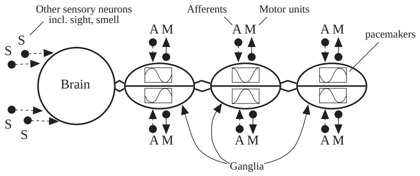


Figure 23: Biological Network

The basic structure consists of a brain at the anterior end of the main nerve chord, followed further down the chord by a series of ganglia – collections of motor and sensory neuron cell bodies. There is generally a ganglion for each segment along the length of the animal, which may house the synchronizing synapses for alternating activity of opposite legs. Interestingly, "even after sagittal ganglionic hemisection a single leg can step rhythmically."[18] Cutting a ganglion in half from the dorsal to ventral along the animal's longitudinal centerline thus further subdivides the node into two groups of neurons with dedicated internal clocks (one per leg). These internal clocks are known to neurobiologists as pacemakers or central pattern generators (CPGs). CPGs are manifest as negative-feedback chemical cycles within neurons which alter membrane properties and change the cell's excitability.

Assuming that ganglia coordinate each leg pair, there must also be a mechanism by which interganglionic signals are passed to result in organized gaits. An analogous system has been studied in the crayfish.[19] Crayfish swimmerets, located ventrally in pairs along the abdomen, beat metachronally. If the abdominal portion of the nerve cord is dissected from the crayfish and isolated from any source of input, a rhythm of activity like the normal one continues in the motor neurons. The activity repeats at about 1.5 Hz, spreading from the rear forward with a delay of about 150 ms between ganglia. The direction of spread suggests that the fifth (posterior) abdominal ganglion drives the others. However, the rhythmic activity persists after removal of the fifth, fourth, and even the third ganglia. Despite the autonomous activity and coordination of these ganglia, the group can also be controlled by neurons located in anterior regions which have axons stretching down into the abdomen. These anterior "control" neurons need not have firing frequencies related to the rhythm of the swimmerets.

Another similar mechanism coordinates the breathing of insects. "Each ganglion of the abdomen and thorax is capable of rhythmic respiratory control in the absence of

[18] ibid. p. 126. from Ten Cata (1928).

[19] Wilson. p. 127. and Herrick, informal observations.

peripheral feedback."[20] Their intrinsic rates are different, but when coupled in the insect the fastest ganglion drives the rest and they oscillate in synchrony.

A scheme like these can easily be imagined for the control of insect locomotion as well. Motor ganglion may be coupled by inhibition or excitation,unidirectionally or reciprocally, and receive control input from anterior locations demanding changes in speed or direction. A loose coupling would allow the fastest ganglion to drive the group under normal conditions. As the excitatory signal propagated forward it would generate the metachronal wave of activity commonly observed. As faster action was required, stronger control signals from the anterior would drive the various segments into closer coordination. Each node CPG would regulate the segment to prevent reciprocal excitations from resulting in runaway frequencies.

One last factor, of obvious importance in the gait adaptations of "odd" amputee insects, is proprioceptive feedback. Proprioception literally means "sensing one's own", and is applied in neurophysiology to the neuronal transduction of information about muscle stretch and motion. Static observations of insects show thatproprioceptive feedback mechanisms act to maintain posture and stability. In at least one species, hair on the legs provides the necessary feedback.[21] Loading the animal does not change its posture until the apparent limit of its muscular resistance is reached, when rapid and complete collapse results. However, when the hairs are removed from the insect's legs its standing height is inversely proportional to the loading (more weight pushes its body closer to the ground). Oscillatory inputs reveal 'phasic reflex effects', of which Wilson treats ones in grasshoppers, locusts, and caterpillars.[22] However, his treatment of the phasic and tonic components of proprioceptive reflexes in cockroaches is much more intriguing:

> A sudden and maintained change of position imposed upon the leg results in greatly altered rate of firing of the motor neurons which then adapt to a new steady rate which is some function of the position. The neuromotor changes resist the imposed movement of the leg. With oscillatory input, the response is also oscillatory at frequencies well above 20 [Hz]. The maximum motor output occurs before the extreme position is reached and the phase of the output is constant over the whole frequency range found in walking (up to 20 [Hz]).[23]

[20] Wilson. p. 127.

[21] ibid. p. 129.

[22] ibid. pp. 129-130.

[23] ibid. p. 130.

The opposite leg of the same segment always responds synchronously and 180 degrees out of phase. Occasionally activity in adjacent segments is also triggered. Such movements are always in phase or 180 degrees out of phase, varying in sign within a given trial. This proprioceptive reflex link between segments is weak and is not likely fundamental to walking coordination. However, feedback within the segment will prove essential to generating complex, “intelligent” gaits.

The cooperative synthesis of the two systems, CPG synchronization and proprioceptive reflexes, that Wilson hypothesized from the knowledge available at that time is consistent with more recent, extensive research on the cockroach byKeir Pearson *et al*. Wilson conceived of a fundamental walking pattern generated in the ganglia, driven from the posterior and frequency-controlled from the anterior. This mechanism would generate the five gaits commonly observed in hexapod walking. Proprioceptive feedback would then adjust basic gaits to maintain posture and adjust to changes in terrain. This could explain gaits of amputees; with legs removed, the remaining legs may have to refrain from protracting at the ‘normal’ time and wait until the next in-tact leg begins to bear some of the load. This would account for the adoption of the diagonal-pairs gait in double-amputee hexapods. Furthermore, variations in grade can be automatically adjusted for, as each leg is required to bear more (retract for a longer period of time) when climbing or less (retract quickly) when descending. The system even deals well with slippage – a slip means that the leg is no longer supporting the body, so other legs wait to protract while the slipped leg protracts quickly to its next footing.

Pearson *et al.* revealed more details of locomotive control in insects. There is a pacemaker for each leg. All six pacemakers are coupled through reciprocal synapses, with the closest coupling between pacemakers of the same segment. “These. . .signals are the neural basis for the magnetic effects, the phase-dependent accelerations and decelerations of the pacemakers.”[24] The coupling coordinates the oscillations of the legs, maintaining functional coherence which is manifest as walking gaits. Reflex feedback from proprioceptive afferents (sensory transduction neurons) adjusts the duration and frequency of motor neuron activity by direct synapse or throughinterneurons. The system which generates and adjusts basic gaits is controlled from a higher level by a single command signal. A weak command signal results in slow, loosely managed gaits. A strong one commands fast, tightly synchronized gaits. The behavioral culmination of this network is, as noted at the outset, phenomenal! Insect walking is purposeful and methodic to a degree, yet it remains simultaneously very adaptive, to the point of

[24] ibid. p. 135.

appearing intelligent. Generating complexity like this from a network of simple mechanisms is so effective in nature that Rodney Brooks *et al* developed subsumption architecture to accomplish the same thing in man-made systems.

This project follows Wilson and Brooks, applying lessons from biology to an experiment in robotics. A series of three processors act as ganglia, each controlling a pair of legs. Switches on the legs provide proprioceptive feedback to the segment's "ganglion", helping to adjust for minor variations in terrain. The ganglia processors are linked by a common electrical "synapse" to the main processor, the brain as shown below:

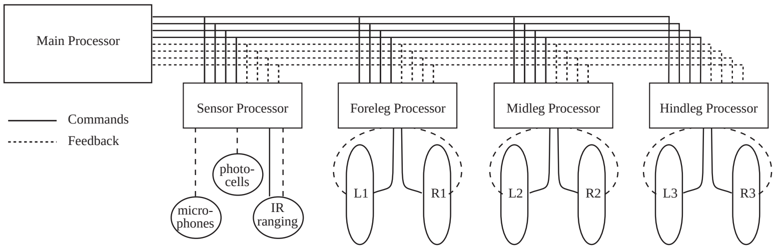


Figure 24: Electronic Network Representation

Signals from the brain request leg movements with simple codes to adjust speed and direction. A single signal may be addressed to one or more processor, allowing synchronization of motion at varying levels of complexity. Finally, there is an "eyes-and-ears" processor which handles IR ranging, visible light sensors, and sonic stimuli. This processor handles the input and reduces it to an above / below threshold message with basic directional information. The main processor accepts messages from the sensor ganglion to incorporate into its control decisions, and in turn may command the ganglion to heighten or reduce its sensitivity to any stimulus. As in the biological structure, basic function responsibility is distributed over a number of subservient systems and coordination is accomplished through behavior fusion at a higher level.

Software:

Subsumption architecture is easily implemented using the C language. Each behavior is implemented as a subroutine. Each behavior subroutine is called in reverse order of precedence (lowest to highest). In a simple system, subsumption may be implemented passively; when an active lower level behavior conflicts with an active higher level behavior, the higher level behavior subsumes the lower level behavior by overriding any conflicts. For example, if *Track Object* has a higher precedence than *Light Search*, and both are activated, *Light Search* will be called first, setting any control bits needed to instruct the control processor to change direction or move. When*Track Object* is called, it would merely reset any of the control bits that are in conflict. Non-conflicting behaviors (or non-conflicting parts of behaviors) are left alone. This "summation" of behaviors is called behavior fusion.

More complex implementations of subsumption architecture can define sub-behavior precedences as well as the more global procedure precedence. Sub-behavior precedences would allow a lower level behavior to subsume some small part of a higher level behavior.

Complex behaviors can be created through behavior fusion. As lower level behaviors are added, a more complex and robust control system will evolve. MuLIR's control system is based upon the more simple passive implementation ofsubsumption architecture. The following behaviors were implemented (in reverse precedence): *Wander*, *Light Avoid*, *Light Search*, *Sound Avoid*, *Sound Search*, *Avoid Object*. Avoid Object has the highest precedence since it is desirable to have the robot avoid running into objects. It is possible for this configuration to cause a behavioral oscillation where the robot moves towards a light / sound source, then has to avoid an object, then attempts to move towards the source again only to have to avoid the same object again, etc. This cannot be helped with such a simple model; however, increasing the number of behaviors would help the robot respond appropriately to such situations by becoming bored, etc.

Implementation

Two major mechanical problems were encountered upon mounting the legs to the chassis. First, the middle leg should be able to fully extend when the front or rear leg is in the same direction. The occlusion plate which triggers the hip limit switch should trigger just before the leg assemblies come into contact. Early tests showed that this was not the case. For both sides of the robot, legs rotating clockwise (at the hip) experienced mechanical interference from the mechanism of the subsequent motor on the same side. This interference is an artifact of the modular leg design process; the original range of motion allowed by the occlusion plates was too great for incorporation into this chassis design. The problem was solved by customizing the occlusion plates to prevent interference :

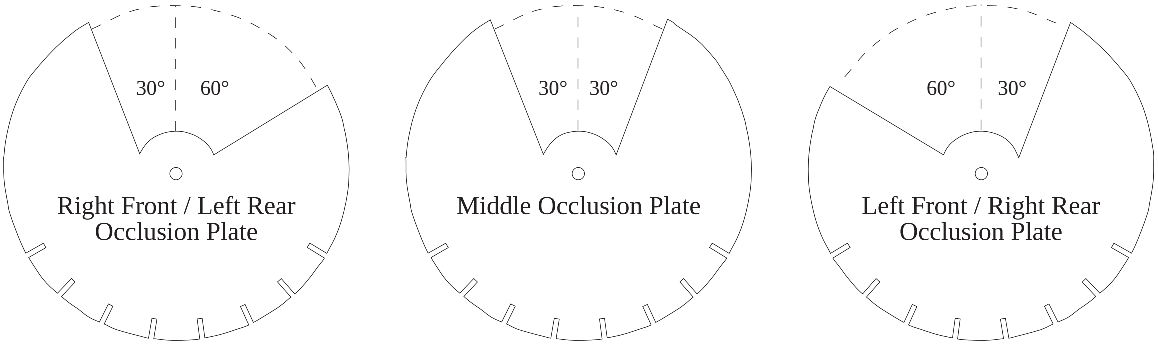

30°
60°
Right Front / Left Rear Occlusion Plate
30°
30°
Middle Occlusion Plate
60°
30°
Left Front / Right Rear Occlusion Plate

Figure 25: Modified Occlusion Plates

This redesign specializes the hardware to some extent, but there are still only three different occlusion plate geometries. The new front occlusion plates have a 60° forward range of motion but only 30° backward range of motion. The middle occlusion plates have a 30° forward and 30° backward range of motion. Finally, the rear occlusion plates have a 30° forward range of motion and a 60° backward range. These new limits prevent the legs from interfering when the legs are fully extended. The new occlusion plates simplify the implementation of walking gaits by ensuring that the limit switches can be triggered and the legs cannot become tangled.

The second problem was due to a failure to calculate the available torque of the motors at the selected voltage. The problem was that the original legs were approximately 8.5 inches in length and the knee motors were unable to lift the robot from a flat starting position. This problem was solved by shortening the legs to approximately 5 inches and running the batteries in series (15 Volts). However this means that only half the current is available, limiting the gaits that may be efficiently implemented. To solve this problem, the legs were shortened further, the foot microswitch's trigger level was curved around the foot, and the foot was rounded to the inside as shown below:

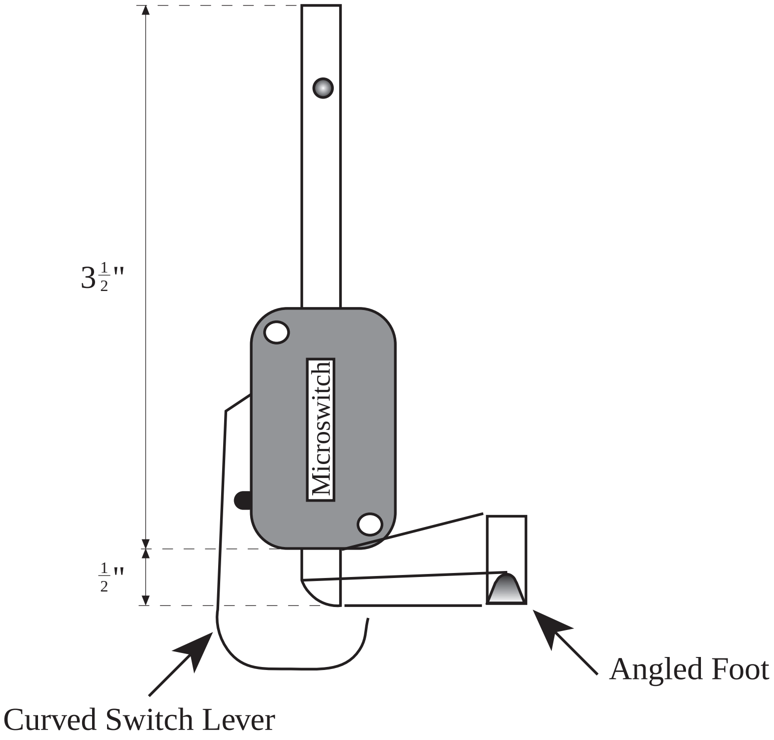


Figure 26: Modified Foot

These factors aided MuLIR in fully supporting its weight even with all legs almost fully extended. The angled foot, suggested by Phil Brandenberger, reduces friction when MuLIR is attempting to stand up. The curved switch trigger also reduces friction and improves balance.

Problems occurred in the wiring due to a distributed ground. Distributed grounds occur when a large current is poured into the ground which causes local peaks in the ground. These peaks cause the ground at one location to differ from the ground at another. The symptoms caused by a distributed ground problem can be quite serious: false interrupts, memory corruption, etc. Observation showed that the problem occurred only when the motors were running. By increasing the current carrying capability from the grounds located near the motors and by attempting to have as common a ground as possible (ie. all ground wires connect in a single location) this problem was minimized.

The code presented in **Appendix A** represents the extent of the tested code at the time of printing. This code should be considered reliable and can be used (as presented) in future projects. The SPI interprocessor communication code was tested using multiple M68HC11EVB boards and the leg control code was also tested on a M68HC11EVB board using the prototype leg. The sensor code was similarly tested and subjected to both real and simulated inputs. A problem with the infrared sensors prevented their testing. The Sharp GP1U52X IR detector looks for light modulated with a 40 KHz carrier centered around 1667 Hz, signals not conforming to these specifications are rejected increasing the signal-to-noise ratio. A LM556 timer chip was used to implement these timing constraints, but the sensors ignored all signals emitted. The output signal was examined with an oscilloscope to confirm that the signal was correct. One problem could be that the LEDs purchased emit light around 940 nm; Sterling Electronics claimed that these LEDs would work with the detector, but another source suggested using 880 nm LEDs (the detector and LEDs are also available from Radio Shack; Sterling Electronics does not provide schematics for the detector). Since the detector would not respond to the IR signal, the IR code was not tested with real signals; however, it was tested using simulated signals.

Results

Time constraints prevented testing MuLIR as a whole. At the time of printing, the subsumption control code had not been tested and some wire wrapping still remained to be completed. However, subsystems were tested and provided encouraging results.

Using the final leg design prototype and a M68HC11EVB board, the leg control software was extensively tested. Several problems, both software and hardware, were identified and corrected. The final control code was extremely fault-tolerant and robust. A hardware problem where the limit switches generated false interrupts was identified and localized, but analysis with an oscilloscope and other similar hardware failed to explain the problem. Finally, the software was changed to ignore interrupts occurring within a fixed time period after movement is initiated; this solution proved quite effective. Next, leg's range of motion was tested and basic walking patterns were tested. From these tests, it was concluded that a complete system would be capable of coordinated walking.

The SPI interprocessor communication system was tested using multiple M68HC11EVB boards. Data rates from 1.5 Mbits / sec to 2.0 Mbits / sec were recorded using a spectrum analyzer. The communication system developed and the software routines can be reliably used in future projects. The SPI system operates in Wired-OR mode to reduce the possibility of processor latch-up in case of a serious programming error.

The completed legs, with modified feet and occlusion plates, were attached to the chassis. An alternating tripod walking gait was examined by manually operating the hip and knee motors. The authors' plan to continue working on the control code and hope to achieve an autonomous system before graduation. In any case, these results are encouraging and indicate that the design was sound and should meet all design goals (with the possible exception of goal 1).

Conclusions

This project was an experience in real world engineering. At the beginning of this project, the envisioned result was quite different from what was actually implemented. For example, the original processor choice was the 8086, this was then rejected in favor of a DSP (for running a complex neural network), this was then rejected in favor of the 6811 due to price-performance considerations. From a more mechanical point of view, many leg prototypes were developed and rejected. Even the final leg design evolved over time to meet all the design goals. Real research and development is a trial-and-error process. Granted, many errors can be avoided with proper research, but some are inevitable.

The completed system should provide an excellent basis for future projects. The hardware and software are highly modular allowing easy extension and modification. The authors hope that students will take advantage of the system and that it will evolve over time. In the following section many suggestions for future projects are made. Some come from the original design, others were conceived along the way. A fully equipped system will provide a robust robot capable of many behaviors with high fault-tolerance.

This project was a realization of a dream long trapped in the fuzzy recesses of the subconscious mind. The greatest thrill was watching the prototype leg behave as instructed by the M68HC11EVB board; once this had been achieved, the rest was but a matter of time. Hopefully anyone with similar dreams will be able to experience the joys of successful robotic development.

Future Projects

MuLIR was designed to be easily modified. The mechanical hardware is modular allowing for changes to be easily introduced. Three of the easiest additions are a top, a coat of paint, and a permanent battery mount to keep the batteries from sliding. The paint is purely cosmetic, but the top may provide greater stability and will definitely offer more protection to the electronics while the battery mount may improve stability. Care should be taken in deciding where to mount the batteries so that MuLIR's stability is maximized and the center of gravity is maintained between the middle legs. More complex additions include adding "whiskers" or "feelers" to the left and right sides of the front plate. The whiskers will reduce MuLIR's chances of running into objects that do not reflect infrared light. Sonar may also be added. The Polaroid Sonar Ranging System interfaces easily with the 6811 and provides excellent distance determination (from 10.5" to over 35' with accuracy to ± ½"). The basic schematic is shown below:

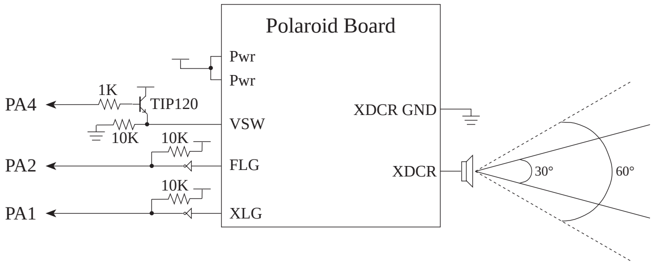


Figure 27: Polaroid Ranging System

The Polaroid Board has six cables which are wired as follows: Yellow and Orange to +VCC, Brown to GND, Red (VSW) to PA4, Blue (XLG) to PA1, and Green (FLG) to PA2. VSW is set high to send a "ping" (to eliminate errors, four pulses are sent at 50, 53, 57, and 60 KHz @ 300V). The Polaroid Board will set FLG high when the ping is sent. Once this happens the code should begin timing. When the return ping is detected, the XLG line will go high. At this time, the system should stop timing. The Distance traveled is then: $D = Scale \cdot \frac{T_f - T_i}{2}$, where T_f - T_i = the total time. Scale is a function of temperature; the speed of sound $c = 331.4\sqrt{\frac{Temp}{273}}$ m/s, where Temp is the temperature in Kelvin.

A Pyroelectric sensor (with Fresnel lens) can be added to allow MuLIR to detect and track humans. Pyroelectric sensors are sensitive to the IR wavelength emitted by humans. The Eltec 442-3 differential pyroelectric sensor with built-in amplifier interfaces directly to one of the 6811's A/D pins. As an object moves across the sensors field, the voltage with swing from positive to negative or vice versa. This voltage swing indicates the objects direction of motion.

Temperature gauges can be added easily from a variety of hardware. Both digital and analog devices are readily available. Strain gauges can be added to the feet to indicate the degree of stability of a foot; interfacing strain gauges to the A/D ports of the leg processors is straightforward. To improve balance and allow MuLIR to navigate uneven terrains, a mercury level sensor could be added. As the angle between the normal to the chassis and ground varies the mercury shifts, allowing current to flow through contacts in the sensor. This information can be used by the control processor to control balance. More complex balancing can be achieved through gyroscopes.

One last hardware sensor addition that would prove most useful is an IR beacon and a power monitor. As power drops, software would cause MuLIR to search out the IR beacon. MuLIR would navigate to the beacon and dock at a recharging base located below the sensor. A simpler design might just have MuLIR detect low power conditions and sound a warning.

A major addition would be to redesign the communication system. Currently, MuLIR uses the SPI system for interprocessor communication. A better system might have a memory-mapped communications system. This would also require rewriting the control software, but the benefits would be quite noticeable.

Further development of the senses is quite possible even without hardware additions. Since subsumption architecture is modular, new behaviors can be added on top of the old behaviors without difficulty. Another possibility would be to rewrite the control code using Fuzzy Logic (which is quite similar in application to subsumption architecture). To increase the biological feasibility, more responsibility could be off-loaded to the leg processors. The would enable MuLIR to operate more efficiently and increase fault tolerance.

An interesting option is to implement multi-tasking in the control processor. Very little code is necessary to implement multi-tasking on the 6811. The system timer is used to measure the time given to any one process; when time has expired the system switches tasks by saving the current task's registers (including the stack pointer) and then reloading the next tasks registers. As long as enough memory is available for multiple stacks, the system is quite simple. *NOTE: There is no memory protection available in the 6811, which could be a problem if the stack overflows!* In addition to the system timer procedure, a minimum of four routines are needed: new_process (PID), kill (PID), usleep (ms), and wait (occurrence). Interactive C implements multi-tasking; the binaries and source code are both available from MIT.

Other projects involve combining previous Senior Design Projects. For example, an interface could be created to control MuLIR from the Virtual Reality system developed by Mark Topinka ('92) and extended by Mark Rieffel ('94). Or a simple implementation of Giray Pultar's ('94) Linear Predictive Coding could be implemented on either the sensor processor or the control processor to have MuLIR recognize basic command words. Other projects are certainly possible, the only limitation being the designer's imagination!

In any addition, fault tolerance should be considered. The Computer Operating Properly (COP) interrupt could be used to ensure that the system is still running correctly. Further precautions should be easy to add.

Acknowledgments

Many people were vital in helping get this project from the inspiration stage to the final stage. We would like to thank our advisors **Silvio Eberhardt** and **Erik Cheever** for their guidance and suggestions as well as the (usually) gentle pressure to keep things moving. **Charlie White** was a great help in obtaining parts from many companies with whom the Engineering Department had not previously dealt. And finally, we would like to thank **Grant Smith** for his invaluable help in the machining process.

Without the help of these people, this project would never have gotten past the fuzzy dream-like strands of thought from which it was conceived. Thank you for your help and time!

Appendix A: Software Control Systems

[NOTE: All code has been removed from this online document. Please contact the author for more information…]

PROTOTYPE LEG CODE BEGINS
PROTOTYPE LEG CODE ENDS

LEG CODE BEGINS

LEG CODE ENDS

SENSOR CODE BEGINS
SENSOR CODE ENDS

Appendix B: Pinout and Wiring Diagrams

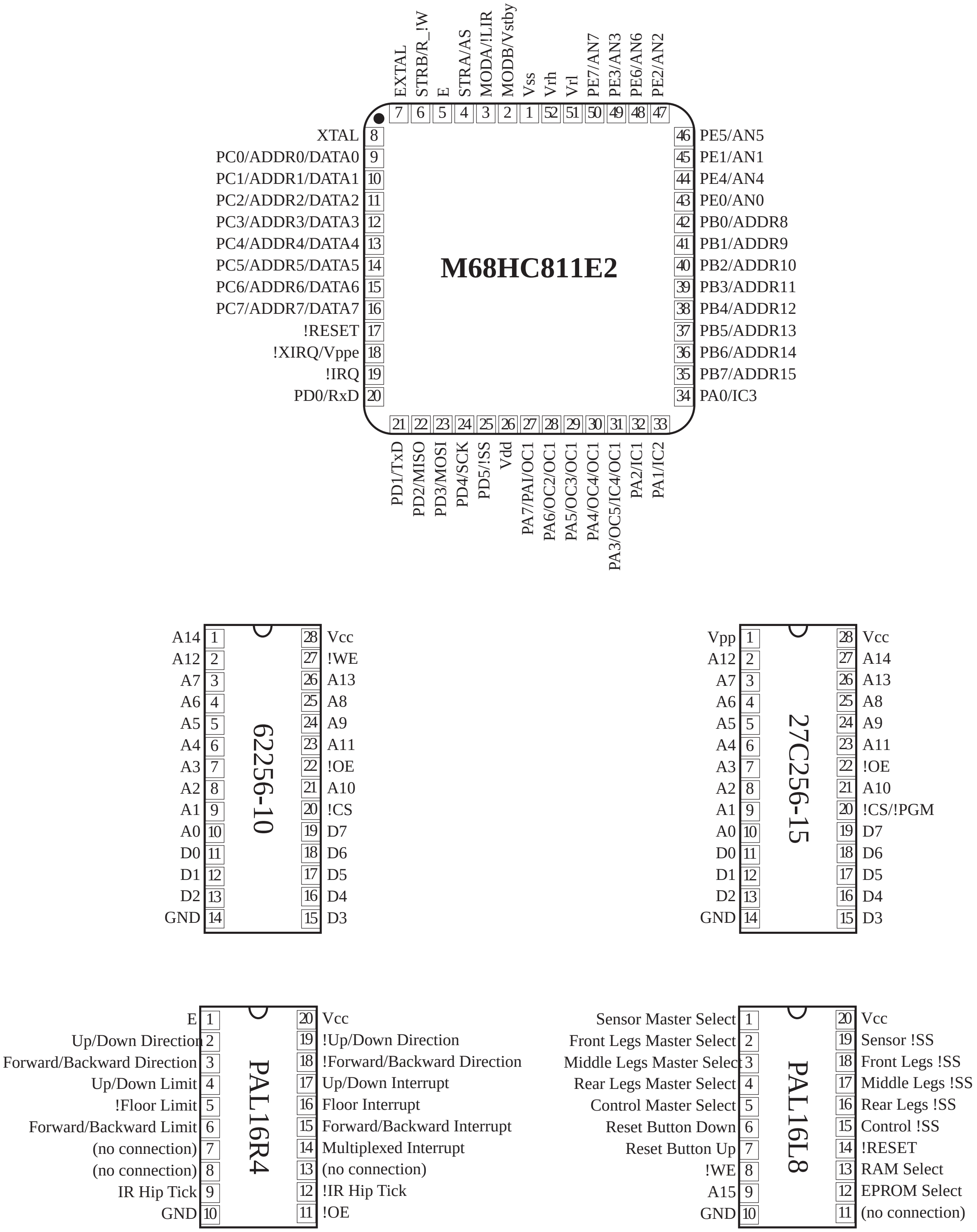


Figure 28: Pinout Diagrams (1 of 2)

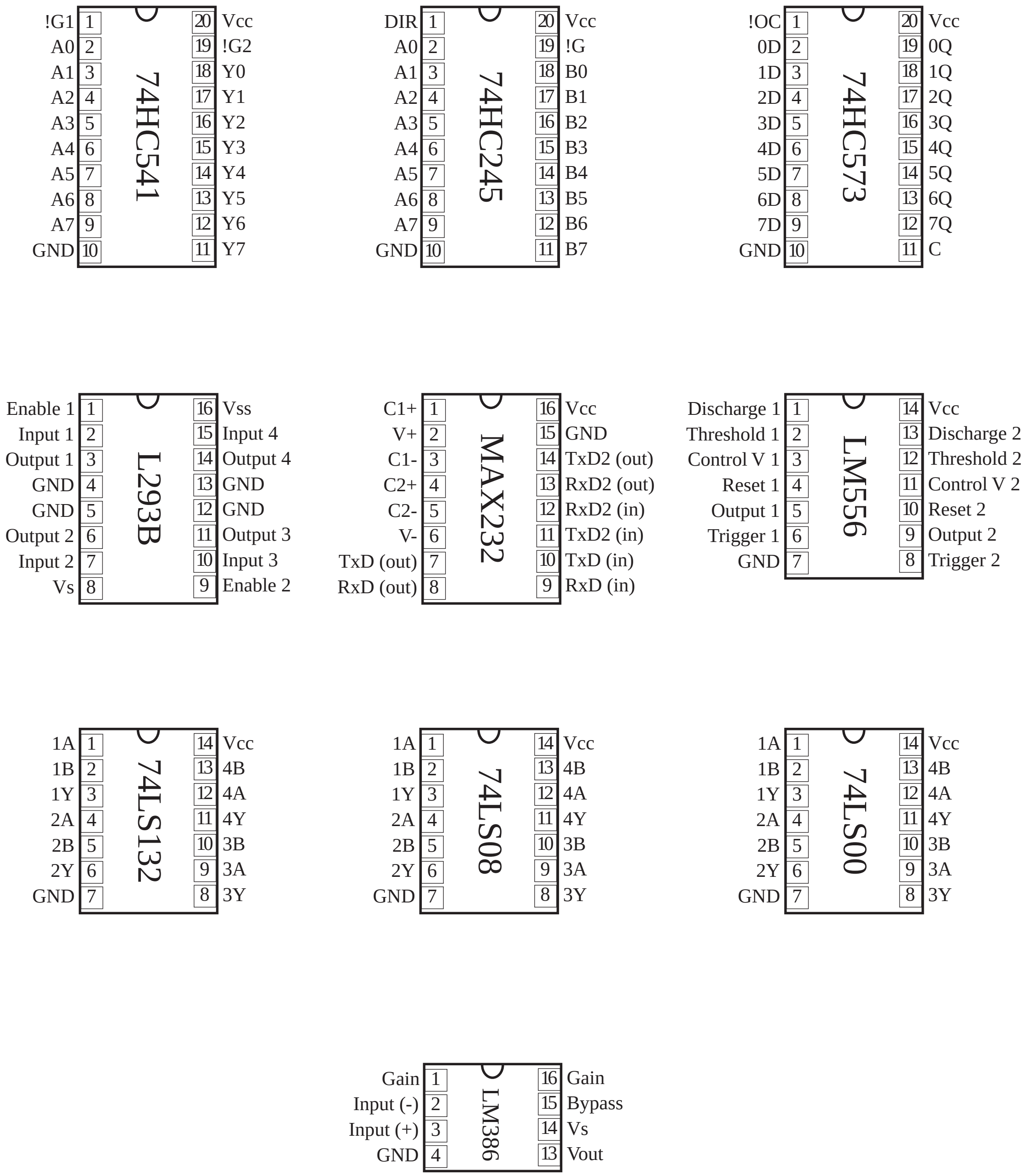


Figure 29: Pinout Diagrams (2 of 2)

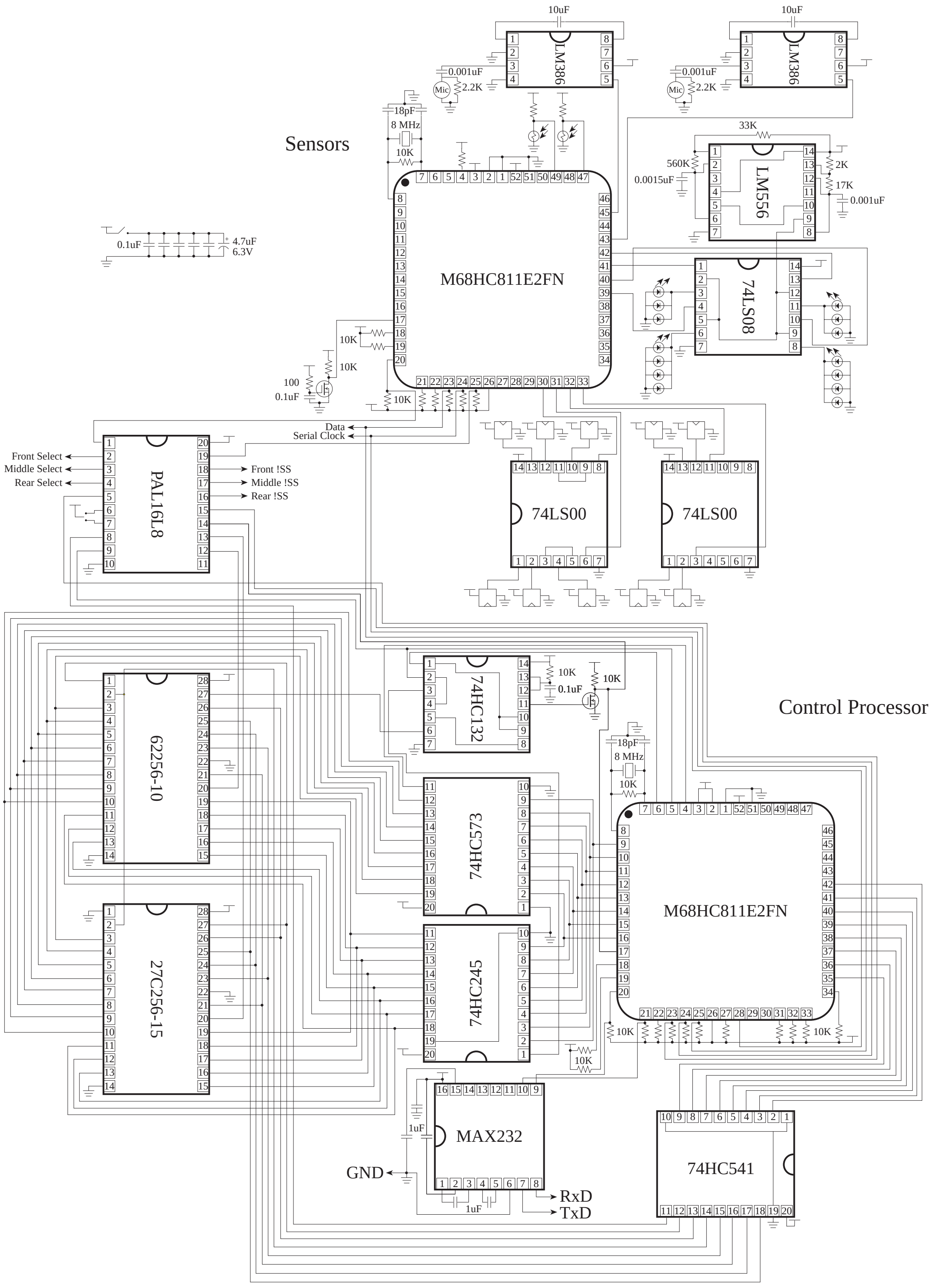


Figure 30: Wiring Diagram (1 of 3)

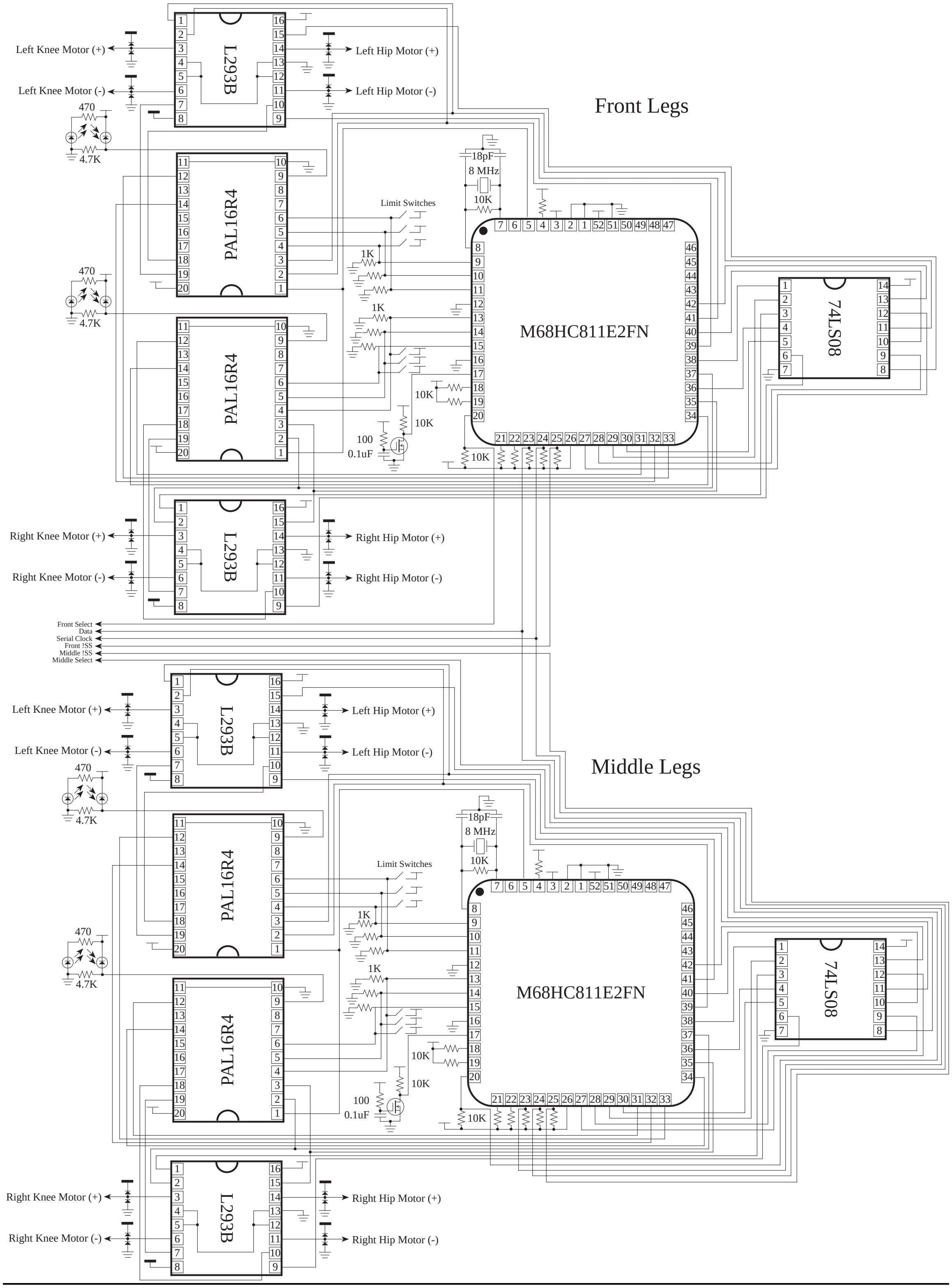


Figure 31: Wiring Diagram (2 of 3)

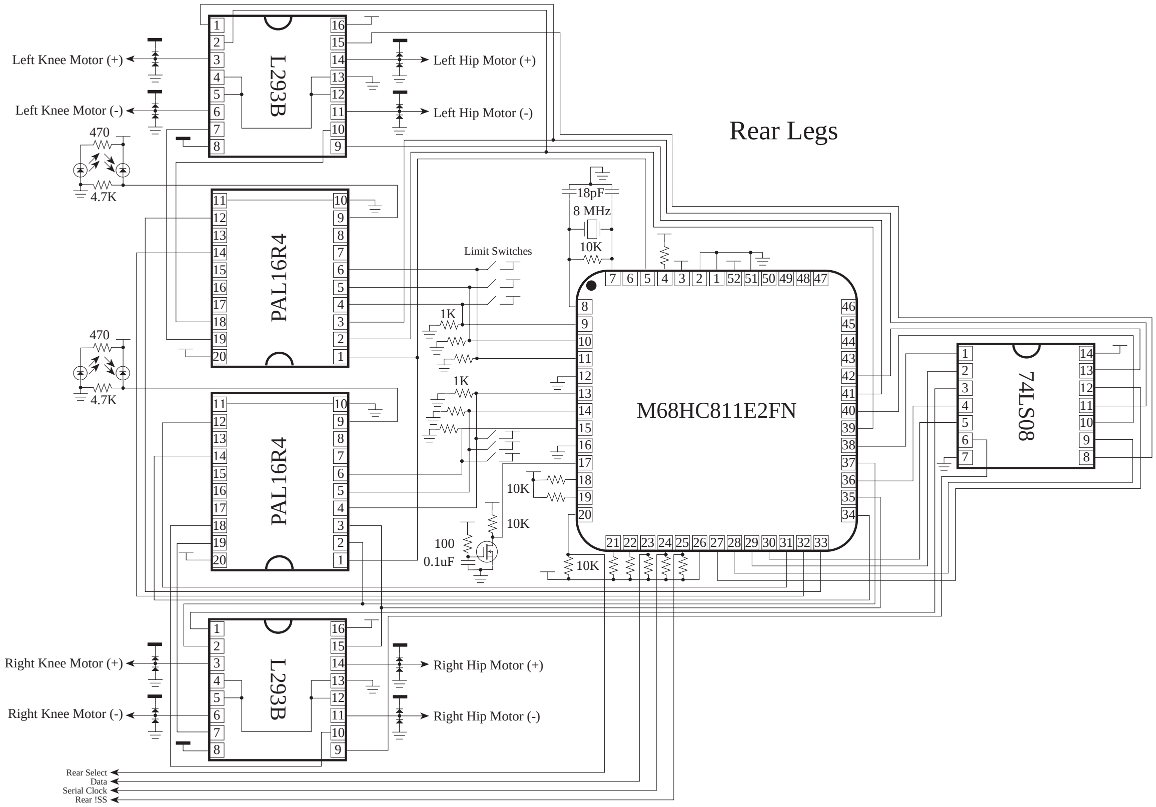


Figure 32: Wiring Diagram (3 of 3)

Pin	Cable Color	Description
1	Black	DB9 Pin 1
2	White	DB9 Pin 2 = Receive Data (RxD)
3	Gray	DB9 Pin 3 = Transmit Data (TxD)
4	Purple	DB9 Pin 4
5	Blue	DB9 Pin 5 = Ground (GND)
6	Green	DB9 Pin 6
7	Yellow	DB9 Pin 7
8	Orange	DB9 Pin 8
9	Red	DB9 Pin 9
10	Brown	(not connected)
11	Black	(not connected)
12	White	(not connected)
13	Gray	(not connected)
14	Purple	(not connected)
15	Blue	(not connected)
16	Green	(not connected)

Table 1: Serial Communications Connector Wiring

Pin	Cable Color	Description
1	Black	Hip Motor (-)
2	White	Hip Motor (+)
3	Gray	IR Receiver GND
4	Purple	IR Receiver Vcc
5	Blue	IR Emitter GND
6	Green	IR Emitter Vcc
7	Yellow	IR Output
8	Orange	Hip Limit Switch Vcc
9	Red	Hip Limit Switch Output
10	Brown	(not connected)
11	Black	Knee Motor (-)
12	White	Knee Motor (+)
13	Gray	Knee Limit Switch Vcc
14	Purple	Knee Limit Switch Output
15	Blue	Foot Switch Vcc
16	Green	Foot Switch GND
17	Yellow	Foot Switch Output
18	Orange	(not connected)
19	Red	(not connected)
20	Brown	(not connected)

Table 2: Leg Connector Wiring

Pin	Cable Color	Description
1	Black	Left Microphone Output
2	White	Left / Right Microphone Ground
3	Gray	Right Microphone Output
4	Purple	(not connected)
5	Blue	Left Photocell Output
6	Green	Left / Right Photocell Ground
7	Yellow	Right Photocell Output
8	Orange	(not connected)
9	Red	Left Front IR Power
10	Brown	Left Front IR Ground
11	Black	Right Front IR Power
12	White	Right Front IR Ground
13	Gray	(not connected)
14	Purple	(not connected)
15	Blue	Left Side IR Power
16	Green	Left Side IR Ground
17	Yellow	Right Side IR Power
18	Orange	Right Side IR Ground
19	Red	(not connected)
20	Brown	(not connected)
21	Black	Left Front IR Detector 1 (from center to outside)
22	White	Left Front IR Detector 2
23	Gray	Left Front IR Detector 3
24	Purple	Left Front IR Detector Power
25	Blue	Left Front IR Detector Ground
26	Green	(not connected)
27	Yellow	(not connected)
28	Orange	(not connected)
29	Red	(not connected)
30	Brown	Right Front IR Detector 1 (from center to outside)
31	Black	Right Front IR Detector 2
32	White	Right Front IR Detector 3
33	Gray	Right Front IR Detector Power
34	Purple	Right Front IR Detector Ground
35	Blue	(not connected)
36	Green	(not connected)
37	Yellow	(not connected)
38	Orange	(not connected)
39	Red	Left Side IR Detector 1 (from front to back)
40	Brown	Left Side IR Detector 2
41	Black	Left Side IR Detector Power

42	White	Left Side IR Detector Ground
43	Gray	(not connected)
44	Purple	(not connected)
45	Blue	Right Side IR Detector 1 (from front to back)
46	Green	Right Side IR Detector 2
47	Yellow	Right Side IR Detector Power
48	Orange	Right Side IR Detector Ground
49	Red	(not connected)
50	Brown	(not connected)

Table 3: Sensor Connector Wiring

Appendix C: Mechanical Diagrams

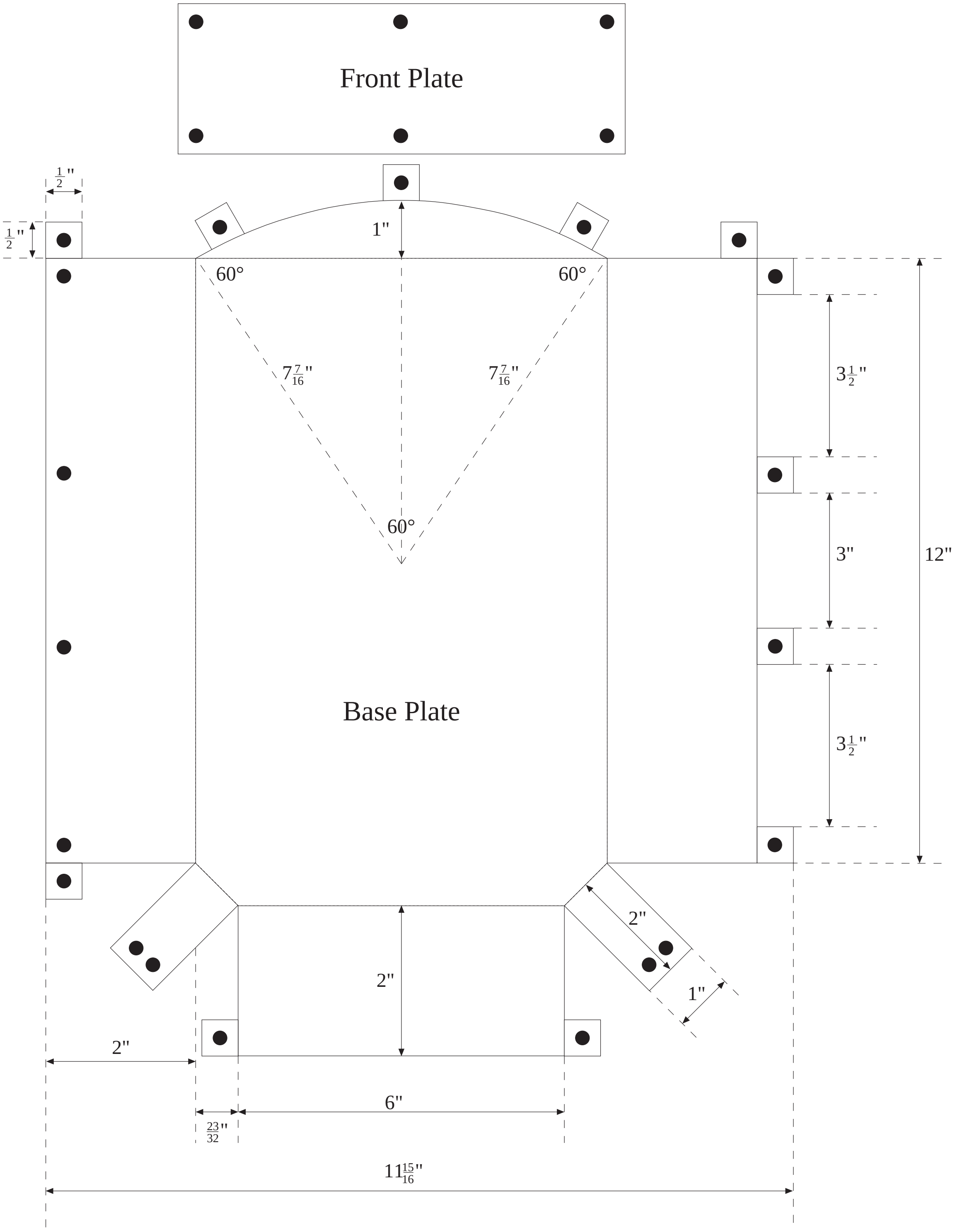


Figure 35: Chassis Design

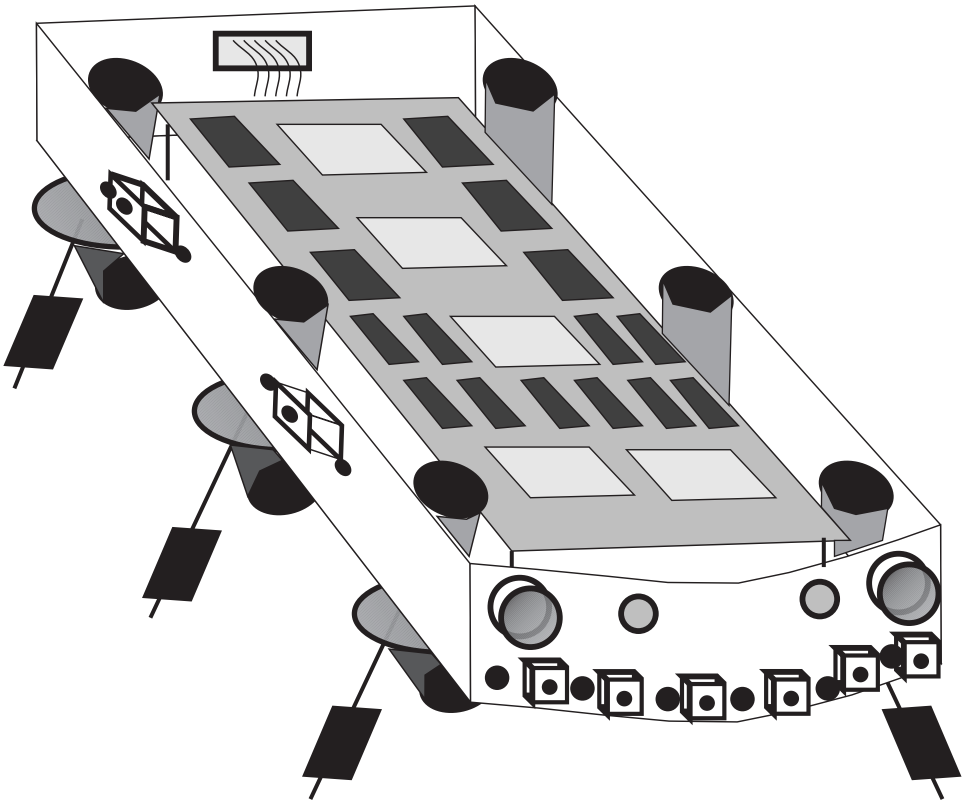


Figure 34: MuLIR

Appendix D: Parts, Manufacturers, and Price List

Company		*Phone Number*	
Quantity	Order Number	Description	Price ($ each)
CGN Company		*(408) 720-1814*	
5	M68HC811E2	Microcontroller	20.00
4	CGN1001	Basic HC11 smart socket	19.00
1	CGN1101-232	RS232/Expanded mode smart socket	30.00
Sterling Electronics		*(800) 745-5500*	
6	L293B	H-bridge integrated circuit	2.38
10	Sharp GP1U52X	IR receiver unit	3.24
15	GL628	940 nm IR LED	0.38
Allied Hobbies		*(610) 639-7247*	
1	Panda Quick Charger	battery charger	89.90
2	Progressive Tech 900	7.5V 1.8A/hr NiCad battery	49.95
3	Tamiya Battery Cable	battery connector	2.75
Digi-Key		*(800) 344-4539*	
5	P308-ND	mini-Red LED	1.68 / 10
1	P309-ND	mini-Green LED	2.30 / 10
1	P310-ND	mini-Amber LED	2.49 / 10
1	3M2802-ND	TEXTOOL Zero Insertion Force IC Socket	17.99
1	SRM20256LC1-ND	256K (32K x 8) CMOS Static RAM (100 ns)	11.63
1	27C256-15/J-ND	256K (32K x 8) CMOS EPROM (150 ns)	6.54
4	DM74LS08N-ND	AND gate	0.58
2	DM74LS00N-ND	NAND gate	0.58
2	LM386N-3-ND	Low Voltage Audio Power Amplifier	1.98
1	LM556CN-ND	Dual Timer	1.65

48	1N4001	50V 1A Rectifier	0.60 / 10
1	BC4AAL-ND	4 AA battery holder	1.06
2	P9931-ND	Electret Condenser Microphone Cartridge	1.34
34	CKN1027-ND	Momentary-Off-Momentary SPDT Toggle Switch	5.91
1	CK4016-ND	On-On SPDT Pushbutton Switch	5.91
4	ED7020-ND	20 pin Strip Socket	1.03
1	CKR14G-ND	14 pin Socket Connector	1.82
6	CKR20G-ND	20 pin Socket Connector	2.14
1	CKR50G-ND	50 pin Socket Connector	3.92
1	CHW14G-ND	14 pin Protected Header	2.49
6	CHW20G-ND	20 pin Protected Header	2.91
1	CHW50G-ND	50 pin Protected Header	5.74
2	ED4308-ND	8 pin Dual-In-Line Wire Wrap Socket	0.92
7	ED4314-ND	14 pin Dual-In-Line Wire Wrap Socket	1.60
6	ED4316-ND	16 pin Dual-In-Line Wire Wrap Socket	1.83
7	ED4320-ND	20 pin Dual-In-Line Wire Wrap Socket	2.29
2	ED4328-ND	28 pin Dual-In-Line Wire Wrap Socket	3.20
1	A2047-ND	DB9 Female Connector	1.45

Newark Electronics *(215) 654-1434*

6	98F027 1873-13	Honeywell Plastic Darlington Opto Switch	2.38

H&R Company *(800) 848-8001*

12	TM92MTR2023	Gearhead Motor 12 VDC 100 RPM	9.95

Swarthmore College Engineering Department

1	62256-10	256K (32K x 8) Static RAM	
2		CdS Photocell	

1	PAL16L8	Programmable Logic Array
8	PAL16R4	Programmable Logic Array
6		470 Ω Resistor
18		1K Ω Resistor
1		2K Ω Resistor
2		2.2K Ω Resistor
6		4.7K Ω Resistor
38		10K Ω Resistor
1		17K Ω Resistor
1		33K Ω Resistor
1		560K Ω Resistor
21		0.001 μF Capacitor
1		0.0015 μF Capacitor
2		10 μF Capacitor
1		On-On 4PDT Switch
1		4" 14 wire Ribbon Cable
1		3' 20 wire Ribbon Cable
1		1' 50 wire Ribbon Cable
4		AA battery
6		Hall Effect Microswitch
19		LED Holder

Glossary

A/D Analog-to-Digital converter. A device which samples analog signals and converts them to a digital value by sampling (the conversion has a fixed resolution).

afferent A primary sensory neuron which transduces a stimulus into an electrochemical signal.

AI Artificial Intelligence. The study of creating machines or computer programs which exhibit signs of higher intelligence.

behavior An action or process which is activated by sensory input.

COP Computer Operating Properly. An periodically generated interrupt which tests if the microprocessor is still operating correctly.

CPG Central Pattern Generator. Neuronal clock or pacemaker able to oscillate in the absence of stimulation.

CPU Central Processing Unit. The sub-system responsible for reading and executing code.

DSP Digital Signal Processor. A processor capable of performing fast operations on digital signals.

EPROM Erasable Programmable Read-Only Memory. Memory that may be programmed and then read. The memory may be totally erased by exposure to ultra-violet light.

EEPROM Electrically Erasable Programmable Read-Only Memory. Memory that may be programmed using high electrical voltages. Lower voltages are used to allow the memory to be read.

ganglion (*pl.* -a) A knot-like mass of nervous tissue consisting of nerve-cell bodies which transmit / receive nerve impulses.

hip The part of the leg mounted to the chassis which allows the leg to rotate forwards / backwards propelling the robot.

IC Input Capture. A 6811 function which generates an interrupt or changes the value on an external pin when an internal state matches a specified state.

Interactive C A 6811 specific implementation of the C language which allows multi-tasking. Interactive C is available via anonymous ftp from the MIT Media Laboratory server cherupakha.media.mit.edu.

interneuron	A neuron which processes and transmits signals among other neurons.
interrupt	A signal which causes the processor to save its state and begin execution of a special section of code. Interrupts are generally used to handle external events efficiently.
IR	InfraRed.
knee	The part of the leg mounted below the robot's body which allows the leg to lift / lower.
LED	Light Emitting Diode
metachronal	Propagating forward (ie. in the direction of motion).
MIT	Massachusetts Institute of Technology
MuLIR	Multi-Legged Interactive Robot.
OC	Output Compare. A 6811 function which generates an interrupt or sets a flag when a specified external state change occurs.
PAL	Programmable Array Logic. A device which may be programmed by electrically blowing fuses to create specific combinatorial logic arrangements.
peristaltic	Of contractions and dilations resulting in forward motion.
proprioception	Literally "sense of one's own." In neurobiology, a sense of muscle tension and motion.
PWM	Pulse Width Modulation. A method for controlling motor speed while conserving energy.
RAM	Random Access Memory. Memory that may be read / written at any location.
register	Special on-chip memory used for holding intermediary data.
sagittal	In dissection, a plane of division defined by a dorsal to ventral and an anterior to posterior line.
SCI	Serial Communication Interface. The 6811 sub-system which allows communication with another computer over a serial link.
SPDT	Single-Pole, Double Terminal Switch.
SPI	Serial Peripheral Interface. The 6811 sub-system which allows interprocessor communication or external peripheral control. The SPI sub-system uses a Master / Slave handshaking protocol which allows only one system to control the SPI data line.

SPST — Single-Pole, Single Terminal Switch.

Subsumption Architecture — A bottom-up programming model which allows behaviors to be easily implemented. Behaviors have priorities, a higher priority behavior will subsume a lower priority conflicting behavior. None conflicting behaviors fuse (behavior fusion) to create new more complex behaviors.

swimmeret — In crayfish, one of a set of small paddle-like appendages located under the abdomen which aid in swimming.

TOF — Timer Overflow. An interrupt generated when the 6811 on-chip timer goes from its maximum 16-bit value, 65525, to 0. For the settings used in this implementation, the timer overflows every 32.768 ms.

UART — Universal Asynchronous Receiver / Transmitter. A 6811 sub-system which controls output from the SCI.

Bibliography

D'Angelo. "The coming of the Robot Ant." *Carnegie Mellon University Alumni Magazine* (July 1992).

Dery, Mark. "Terminators." *Rolling Stone* (June 10, 1993).

Dewdney, A. K. "Insectoids Invade a Field of Robots." *Scientific American* (July, 1991).

Freedman, David H. "Invasion of the Insect Robots." *Discover* (March, 1991).

HC11: M68HC11 E Series: Technical Data. Motorola, 1993.

International Joint Conference on Neural Networks: Volume I. IEEE, 1992.

Jones, Joseph L. and Anita M. Flynn. Mobile Robots: Inspiration to Implementation. AK Peters, 1993.

Krotkov, E. P., R. G. Simmons, and W. L. Whittaker. "Ambler: Performance of a Six-Legged Planetary Rover." International Astronautical Federation, 1992.

Ludeman, Lonnie C. Fundamentals of Digital Signal Processing. John Wiley Sons, 1986.

Machine Intelligence and Autonomy for Aerospace Systems. Edited by: Ewald Heer and Henry Lum. American Institute of Aeronautics and Astronautics, 1988.

M68HC11: Reference Manual. Motorola, 1991.

McComb, Gordon. The Robot Builder's Bonanza. Tab Books, 1987.

Monastersky, Richard. "The Inferno Revisited." *Science News* (Volume 141).

Poole, Harry H. Fundamentals of Robotics Engineering. 1989.

The Robot's Dilemma. Edited by: Zenon W. Pylyshyn. Ablex Publishing, 1987.

Raibert, Marc H. Legged Robots That Balance. MIT Press, 1986.

Robot Design Handbook / SRI International. Edited by: Gerry B. Andeen. 1988.

Robot Motion: Planning and Control. Edited by: Michael Brady [et al.]. 1982.

Rosenthal, Donald A. and Mark D. Johnston. "A System Architecture for a Highly Autonomous Mars Rover." International Astronautical Federation, 1992.

Schondorf, Steven Y. and John E. Gilbert. "Design of a Five Kilogram Mars Micro-Rover." International Astronautical Federation, 1992.

Selverston, Allen I. "A consideration of invertebrate central pattern generators as computational data bases." *Neural Networks, Volume 1*. (1988).

Song, Shin-Min and Kenneth J. Waldron. Machines that Walk: The Adaptive Suspension Vehicle. 1989.

Theory and Practice of Robots and Manipulators: Proceedings ofRoManSy '84, the Fifth CISM-IFToMM Symposium. Edited by: Morecki, Bianchi, Kedzior. 1985.

Todd, David J. Walking machines: An Introduction to Legged Robots. 1985.

Wallich, Paul. "Silicon Babies." *Scientific American* (July 1991).

Wilson, Donald M. "Insect Walking" in The Organization of Action: A New Synthesis. Edited by C.R. Gallistel. Lawrence Erlbaum Associates, 1980.

Wolkomir, Richard. "Working the bugs out of a new breed of 'insect' robots." *Smithsonian* (June 1991).

Index