# Emergent Behavior Robot

**Senior Project Final Report**

**By:**

**Andrew Elliott & Nick Hanauer**

**Project Advisors:**

**Joel Schipper & Nick Schmidt**

**Bradley University**

**Department of Electrical and Computer Engineering**

**Peoria, IL**

**May 14, 2010**

## *Abstract*

Emergent behavior results when many small, simple behaviors combine to produce a larger, more sophisticated behavior. The goal of this project is to design and build a robot platform from the ground up and program it to display emergent behavior. All parts were selected and hardware designed to match the specifications set by the desired behaviors for Emergent Behavior Robot (EBR). EBRs integrated components include infrared range finding sensors, pushbutton bump sensors, microphones, and reflective light sensors. The software is written in C and the ATmega128 microcontroller is programmed with WinAVR. Currently, EBR operates using a system of states that include the following behavior modules: obstacle detection and avoidance, sound avoidance, and reaction to shade of floor tiles. Although in its present state there is no emergence, EBR successfully maneuvers around its environment responding to stimuli.

# *Table of Contents*

# *Table of Figures*

# *Introduction*

The objective of this project is to design and build a robot that uses a software architecture based on subsumption to demonstrate emergent behavior. Subsumption architecture is an approach for designing intelligent systems based on several behavior modules [1]. The idea of emergent behavior is that multiple simple behavior modules combine to create a sophisticated, intelligent response that is greater than the sum of the parts [2]. The first step was to build the platform for Emergent Behavior Robot (EBR). Parts were chosen based on the specifications required for the behaviors to be demonstrated by EBR. These simple behaviors include obstacle avoidance, detection of loud sounds, favoring dark floor tiles, and searching for a beacon. All components are controlled by software and integrated together to give EBR functionality.

The success of this project depends on the hardware-software interface to correctly display an emergent behavior in response to EBR's surroundings. The environment serves as the inputs into the system and dictates how EBR reacts. EBR is not deterministically programmed to perform specific tasks, but rather to perform many small, unrelated behaviors, which contribute to an emergent behavior.

This paper discusses the structure of the hardware with detailed explanations of each component, the current software that has been developed, the desired software to be developed, and finally the results and conclusions from this project.

# Hardware

Figure 1 is a high-level block diagram that shows the various subsystems EBR uses to sense and react to its environment. The robot platform was built from the ground up to ensure that the design has the features necessary to execute its behaviors successfully. The platform is controlled by an ATmega128 microcontroller, which is programmed in C. The microcontroller utilizes the inputs from several sensors to extract the environmental features necessary to react intelligently.

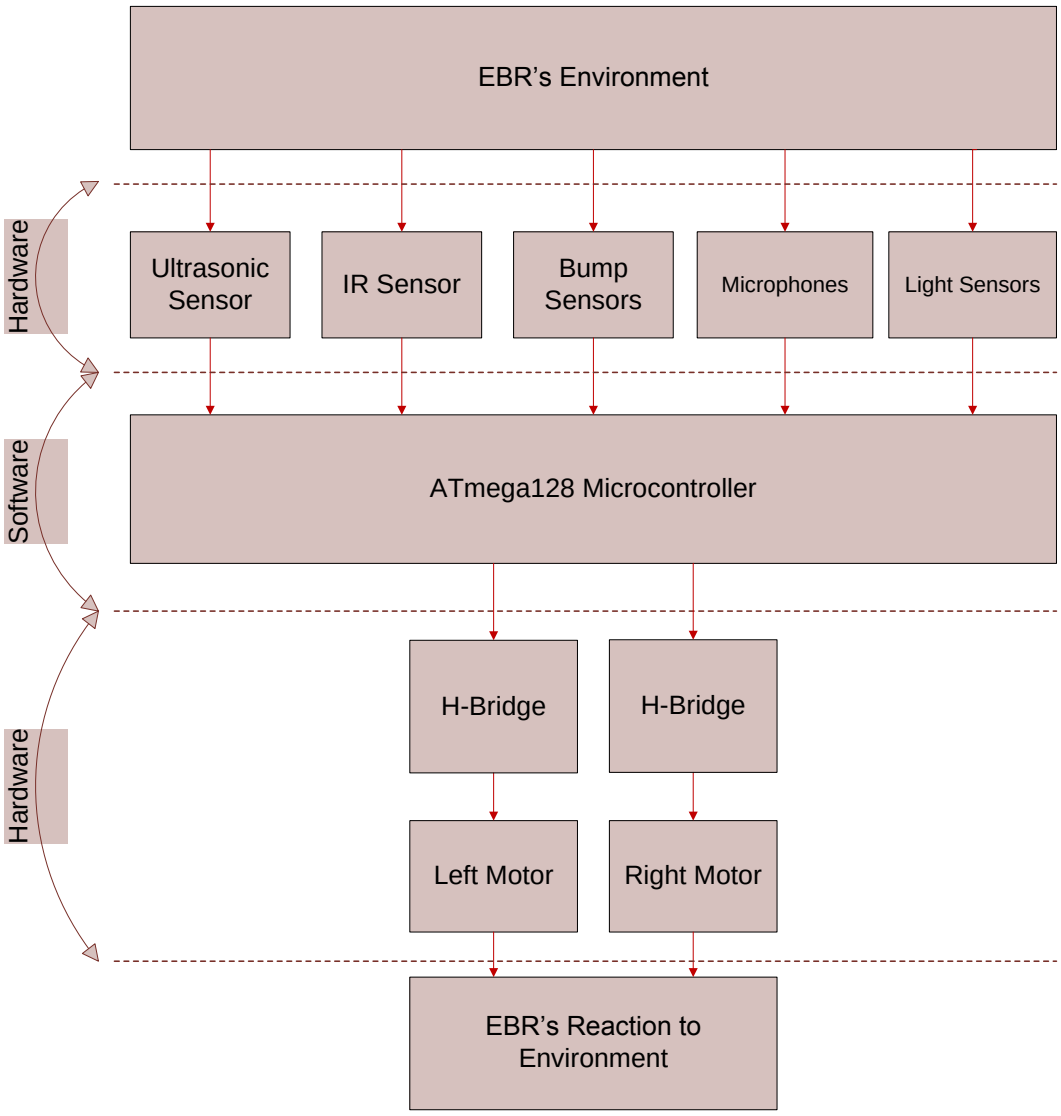



**Figure 1: High-Level System Block Diagram**

EBRs behaviors were decided on, then sensors were found to implement those behaviors. While searching for an ultrasonic transmitter EBR will perform obstacle avoidance and react to noise. When EBR has been affected by a loud noise EBR will begin to search for a dark tile. EBR uses multiple different components to perform these behaviors, and each of these components will be explained in further detail below.

Each hardware component mentioned here is explained in further detail in the following sections and in the order mentioned here.

**Ultrasonic Receiver/Transmitter**

One behavior EBR performs is locating an ultrasonic beacon. A paired 40 kHz ultrasonic receiver, 40TR16F, and transmitter, 40TR16P, were selected. During testing, it was discovered that the ultrasonic receiver could only read a signal from approximately 70cm away. Further testing using the transmitter on the SRF05 ultrasonic ranger provided two possible reasons for the short range. First, it was noticed that by increasing the amplification on the ultrasonic receiver, the detection distance was increased. However, the amplifier was already hitting the 5V rail so no further amplification of the signal could be done. A second possibility is a phase difference between the ultrasonic transmitter on the SRF05 and the receiver built. This component was not interface with software because of the issues encountered. For the ultrasonic receiver circuit see Figure 17 in Appendix A.

**Infrared Sensors**

The Sharp GP2D12 was selected for obstacle avoidance because of its operating range. This sensor has a range of 10cm-80cm. The IR sensor range is important for our application because EBR needs to detect and avoid objects in its path of travel. Figure 2 is a picture of the IR sensor before it was mounted on EBR. To see the pin connection between the IR sensors and the microcontroller see Appendix A, Figure 14.

**Figure 2: GP2D12 Infrared Sensor**

### Bump Sensors

The bump sensors were added as a secondary means of obstacle detection. Simple pushbuttons were mounted in an array on the front of EBR. The main requirements for the pushbuttons were to have a raised actuator for easy mounting, and be a momentary switch so the switch was only active when an object is in contact with EBR. The pushbutton chosen has a .33cm actuator, one of the largest readily available. To attach the bump sensors to EBR, a bracket was constructed and fixed to EBR. We also constructed bumper bars, seen in Figure 3, to make a larger surface area. To see the pin connection between the bump sensor array and the microcontroller see Figure 14 under Appendix A.

**Figure 3: Bump Sensor Array**

**Microphones**

A set of four microphones are used to detect loud noises and determine the direction of the source of the noise. A microphone is located in each corner of the chassis, and the microphone that triggers first will provide the general direction of the source of the sound. The electret microphones selected could sense sounds louder than 80dB between 20Hz and 20 KHz, which is the audible range. The 80dB is important for our application because we defined a loud noise as being above 80dB. The microphone outputs were amplified and filtered. Amplification allowed the microcontroller to read the output and a low pass filter was added to eliminate undesired noise. The low pass filter is set at 7.2 KHz to cut out high frequency noises that could be caused by the system. This frequency is also chosen because the PWM is run at 7.8 KHz and could affect the microphones. An image of the microphone circuit, including the amplifier and low pass filter, can be seen in Figure 4. The microphone circuit diagram can be seen in Figure 16. The calculations for the low pass filter can be seen in Calculation 1 and Calculation 2. The calculation for the amplifier is in Calculation 3. Calculation 5 shows the speed of a microcontroller required to determine directionality of a noise.

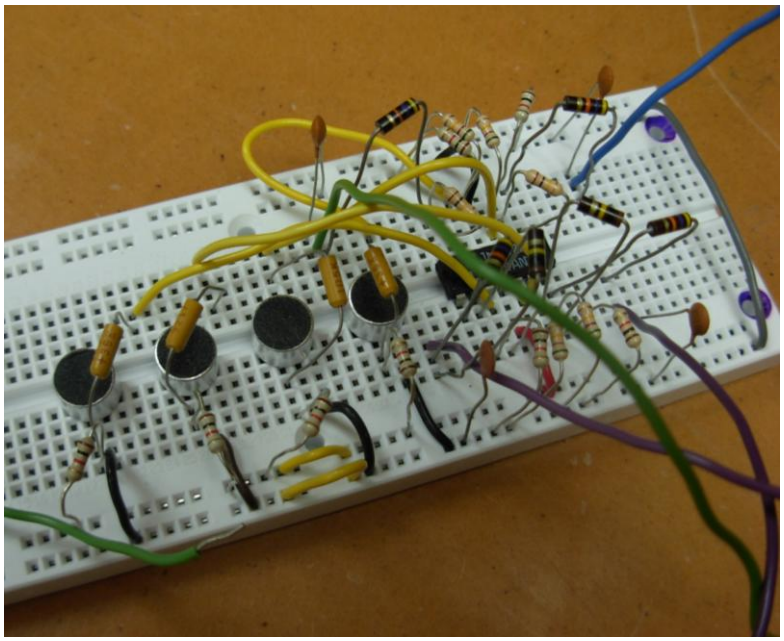


**Figure 4: Microphone Circuitry**

### Reflective Light Sensors

In our application reflective light sensors determine the shade of the floor tiles. EBR is programmed to prefer the darker tiles, and it tries to seek out the darker tiles after it has detected and is avoiding a loud sound. The problem with most reflective light sensors is that they are designed to be close to the object they are sensing. To ensure that the reflective light sensor could read the floor tiles the sensor needed to be able to read approximately 1.9cm, which is the distance from the bottom of EBR to the ground. The chosen sensor operating range is between 0.5 and 2.54cm. Figure 5 is an image of the reflective light sensor. To view the reflective light sensor circuitry, see Figure 18 and to view the resistor calculation for the LED see Calculation 4.

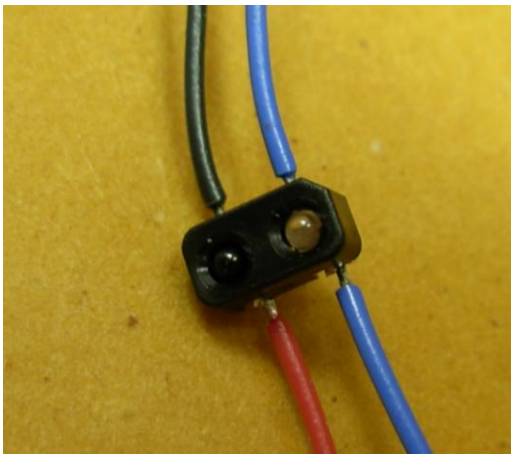


**Figure 5: Reflective Light Sensor**

### Microcontroller

The first component provided was the BDMICRO MAVRIC-IIB. This microcontroller board, based on the ATmega128 MCU, was the first component selected. It has a 16 MHz clock, 128K Program FLASH, 4K Static RAM, 4K EEPROM, three configurable timers, six high resolution PWM outputs, eight channel 10-bit A/D converter, up to 51 I/O pins, two UARTs, 5V regulator, and more. The connection between the microcontroller and other hardware can be seen in Figure 14.

## H-Bridges

The only way EBR responds to its environment is through movement. The H-bridges allow the microcontroller to control rotational speed and direction of the motors independently. H-bridges were chosen to match the motors. The H-bridges needed to be able to handle the 2.8A stall current of the motor. Our choice was the H-Bridge component developed by Acroname based on the LMD18200 H-Bridge. The LMD18200 module is capable of handling 3A continuous and 6A surges. The H-Bridge component can be seen in Figure 6. For the motor and H-bridge circuitry see Figure 15.

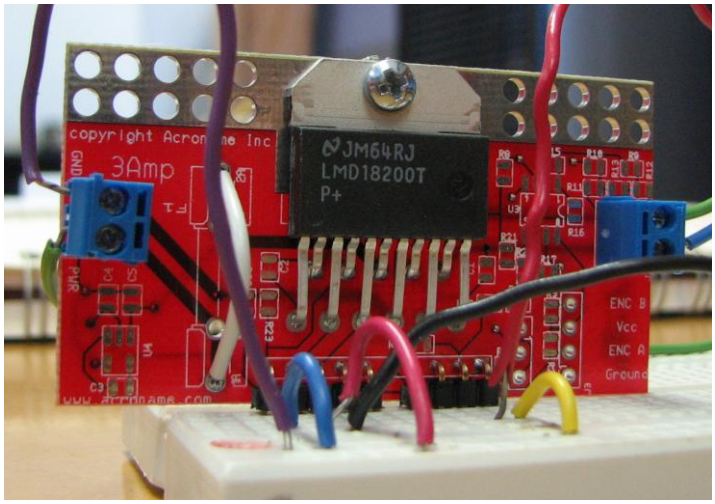



**Figure 6: LMD18200 H-Bridge**

## Motors

The motors are required to drive EBR at a maximum speed of 122cm/s, which is approximately twice the normal walking speed. To meet this specification the motor RPM must be higher than 305. This minimum speed was determined based off travelling 122cm/s using 7.62cm wheels. Also, the motors needed to have a torque high enough to move a ten-pound robot. The torque required to accomplish this is 38.1 cm*lbs, which will move a ten-pound robot 3.81cm. The motor selected was the BHG31 permanent magnet geared motor which has an RPM of 360 and a torque of 62.7 cm*lbs. The motors run off 6v-24v DC power and have a gear ratio of 31:1. The BHG31 motor mounted to the chassis can be seen in Figure 7. For the motor and H-bridge circuitry see Figure 15.

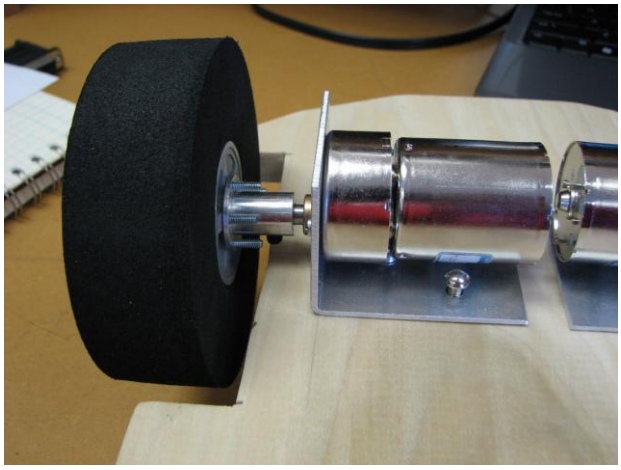


**Figure 7: Motors Mounted on Chassis**

## Chassis

EBR is designed to run in a classroom environment, so a small, lightweight chassis was desired. It is driven by two wheels and has one caster for stability. The drive wheels are mounted near the front of EBR and the caster is mounted in the rear. The caster is in the rear of EBR because it is easier to drag a caster than to push one. A 3-dimensional rendering of the chassis made in GoogleSketchUp started the design process. Once the chassis was designed, it was cut from wood. The 3D rendering and actual chassis can be seen in Figure 8. The final chassis construction can be seen in Figure 9.

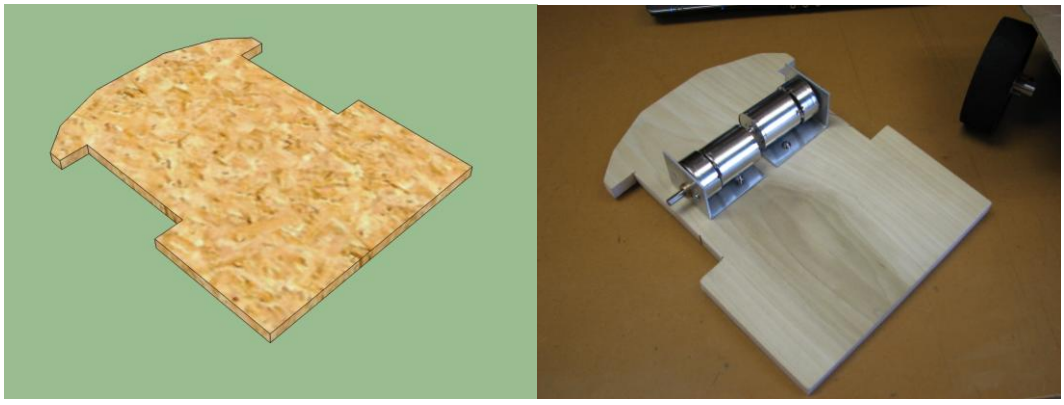


**Figure 8: 3-Dimensional Rendering and Actual Chassis**

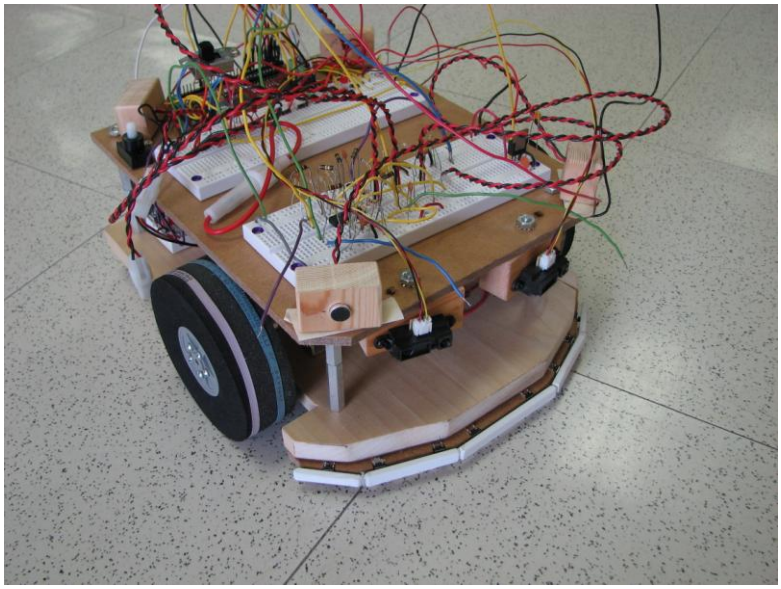


**Figure 9: Final Chassis Construction**

### Wheels

The wheels selected were constructed of lightweight foam with plastic hubs. Additional aluminum hubs were attached to simplify mounting the wheels to the motor shaft. The wheels have a diameter of 7.6cm, which allows EBR to travel faster than the specified speed of 122cm/s. Figure 10 shows a wheel with the attached aluminum hub.

**Figure 10: Wheels and Hubs**

**Batteries**

The motors require 24V, while all other circuitry runs off 5V. We chose to use two 12V 2300mAh Nickel-Metal Hydride batteries seen in Figure 11. The batteries are connected in series to produce 24V for the H-Bridge/Motor circuitry. A regulator connected to one of the batteries generates the 5V for all other circuitry. The regulator circuit can be seen in Figure 19 and the connection to the microcontroller can be seen in Figure 14. Unevenly distributing the circuitry drains one battery much faster than the other, making the battery under heavier load limit the amount of time EBR can run.


**Figure 11: Batteries**

## *Software*

As each hardware component was completed, it was interfaced individually with the microcontroller. Software was developed to test each component and create a structure that allowed for easy integration once all hardware components were added to EBR.

The current software implements obstacle detection and avoidance, sound detection and avoidance, and response to the shade of the floor tiles. The software causes EBR to respond to each input individually, therefore the resulting response is not emergent. Figure 12 shows the flowchart for the current software structure.

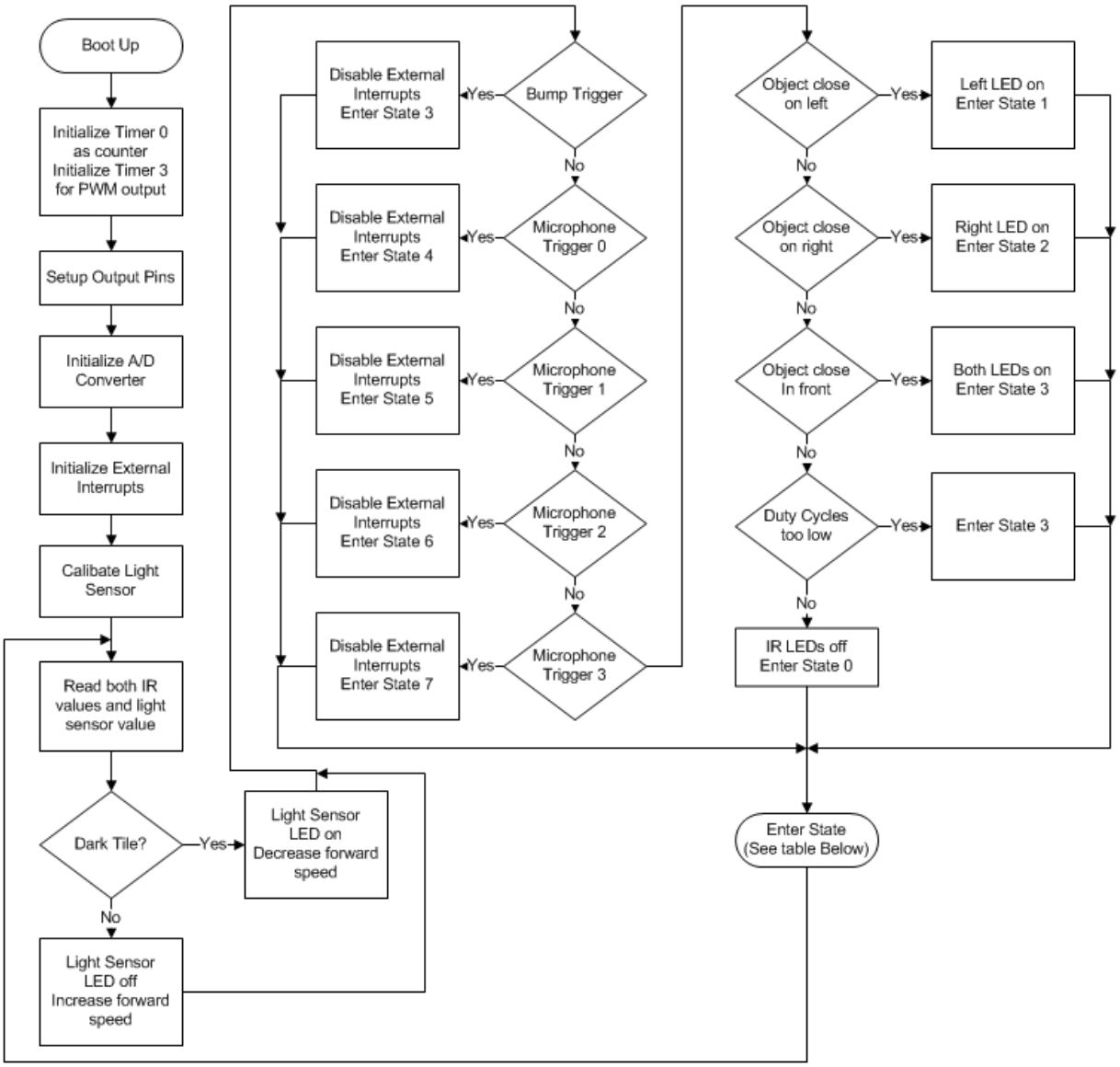



**Figure 12: Current Software Flowchart**

The software starts by setting up timers 0 and 3. Timer 0 is used to generate a general purpose one millisecond counter from the output compare interrupt feature. Timer 3 produces the two 7.8 kHz PWM outputs used to control the motors. All of port B on the ATmega128 is used as outputs to drive LEDs for debugging. The pin outs for the LEDs can be seen in Figure 14. Other outputs configured include the PWM and direction pin outputs to the H-bridges and the analog-to-digital converter (ADC) for reading various sensors. Finally, the external interrupts are configured to trigger on a rising edge.

The reflective light sensor is connected to the ADC. It is calibrated when the software is started. For proper calibration, EBR should be on a white tile when turned on. The current software uses the information from the reflective light sensor to travel at 38.5cm/s, half the normal speed, when EBR is on top of a darker tile. Normal travel speed is approximately 77.7cm/second.

The next step in software is to check if any interrupts have set a flag. The bump sensor array and each of the microphones are connected to an external interrupt. If one of the interrupt flags is set, the software transitions to the state corresponding to that specific interrupt. If no flags are detected, the infrared sensor values are checked. If an object is too close, the software responds by entering the associated state. A list of all the states can be found in Table 1. The default state implements obstacle avoidance by slowing the wheel opposite the obstacle. The wheel is slowed down proportional to how close the object is to EBR.

**Table 1: Current Software States**

| State | Situation | Action |
|---|---|---|
| 0 | No triggered events | Drive forward, adjust duty cycle to avoid close objects |
| 1 | Object too close on left | Reverse for 250ms, rotate clockwise for 500ms, go to state 0 |
| 2 | Object too close on right | Reverse for 250ms, rotate counter-clockwise for 500ms, go to state 0 |
| 3 | Object too close in front/Bump sensor triggered/Duty cycles too low | Reverse for 1000ms, rotate counter-clockwise for 700ms, (if bumped) clear interrupt and turn off Bump LED, go to state 0 |
| 4 | Microphone 0 triggered (front left) | Reverse for 500ms, rotate clockwise for 700ms, clear interrupt, turn off Microphone 0 LED, go to state 0 |
| 5 | Microphone 1 triggered (front right) | Reverse for 500ms, rotate counter-clockwise for 700ms, clear interrupt, turn off Microphone 1 LED, go to state 0 |
| 6 | Microphone 2 triggered (rear left) | Rotate clockwise for 300ms, clear interrupt, turn off Microphone 2 LED, go to state 0 |
| 7 | Microphone 3 triggered (rear right) | Rotate counter-clockwise for 300ms, clear interrupt, turn off Microphone 3 LED, go to state 0 |

Currently, the developed software does not exhibit an emergent behavior. To develop the emergent software, EBR's response to all stimuli must be simultaneous. The desired result from the emergent software would cause EBR to respond intelligently to its environment. The signals sent to the H-bridges would be calculated using a weighted equation structured similar to the following equation:

$$duty\_cycle = \omega_0(roam) + \omega_1(obstacle) + \omega_2(sound) + \omega_3(tile) + \omega_4(beacon) \qquad (1)$$

The weights would be adjusted based on the environment and the priority levels assigned to each behavior. A high priority behavior would always have a higher weight than a lower priority one. This way the high priority behavior influences the response more. The priority levels can be seen in Table 2.

**Table 2: Behavior Priority Levels**

| **Task** | Obstacle Avoidance | Avoidance of loud sounds | Beacon found | Travel on dark tiles | Roam |
|---|---|---|---|---|---|
| **Priority** | 1 | 2 | 4 | 5 (3 when evading sound) | 6 |

Figure 13 is an example situation EBR could encounter. Independently, each behavior would cause EBR to respond in a particular way. The combined response is difficult to predict, but would be a summation of the independent reactions, using the weighted equation, resulting in an intelligent emergent reaction.

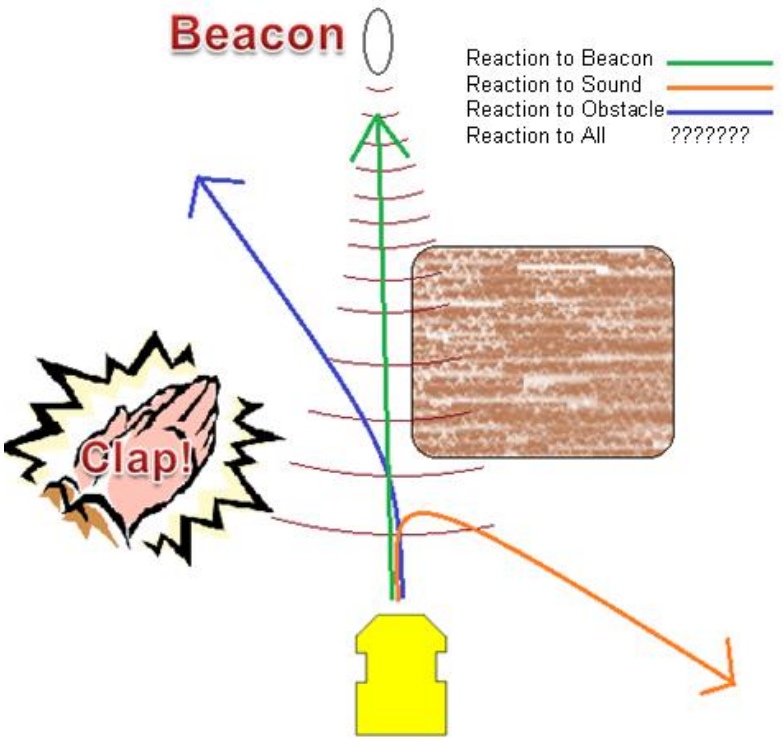



**Figure 13: Potential Situation EBR Could Encounter**

## *Results and Conclusion*

We were able to design and build a robot platform from the ground up. EBR successfully uses IR sensors and bump sensors to detect and avoid obstacles. As EBR approaches an object, the microcontroller dynamically changes the duty cycle of the motors to change direction. Once an object is in contact with EBR the bump sensors detect that EBR needs to reverse, change directions, and then continue going forward. The microphones allow EBR to correctly detect the general direction of a loud noise around EBR and respond by traveling away from the source. Also, EBR uses the reflective light sensor to distinguish between dark and light tiles on the floor. When EBR is travelling over dark tiles the speed reduces to half that of when EBR is travelling on white tiles. More time would have allowed for further development of the emergent behavior software.

The ultrasonic sensors and beacon were not integrated due to time constraints, but test circuitry was created for the receiver. Although the ultrasonic receivers were expected to read the ultrasonic transmitter from approximately 305cm, the sensors are only capable of reading 71cm. Further development of a transmitter could provide a farther distance for the ultrasonic beacon and allow it to be integrated onto EBR.

# *Appendix A: Circuitry Figures*

Appendix A contains the circuit diagrams for all circuitry built on EBR. There is a basic block diagram showing pin outs of the Atmega128 connected to different components as well as the different component diagrams themselves.

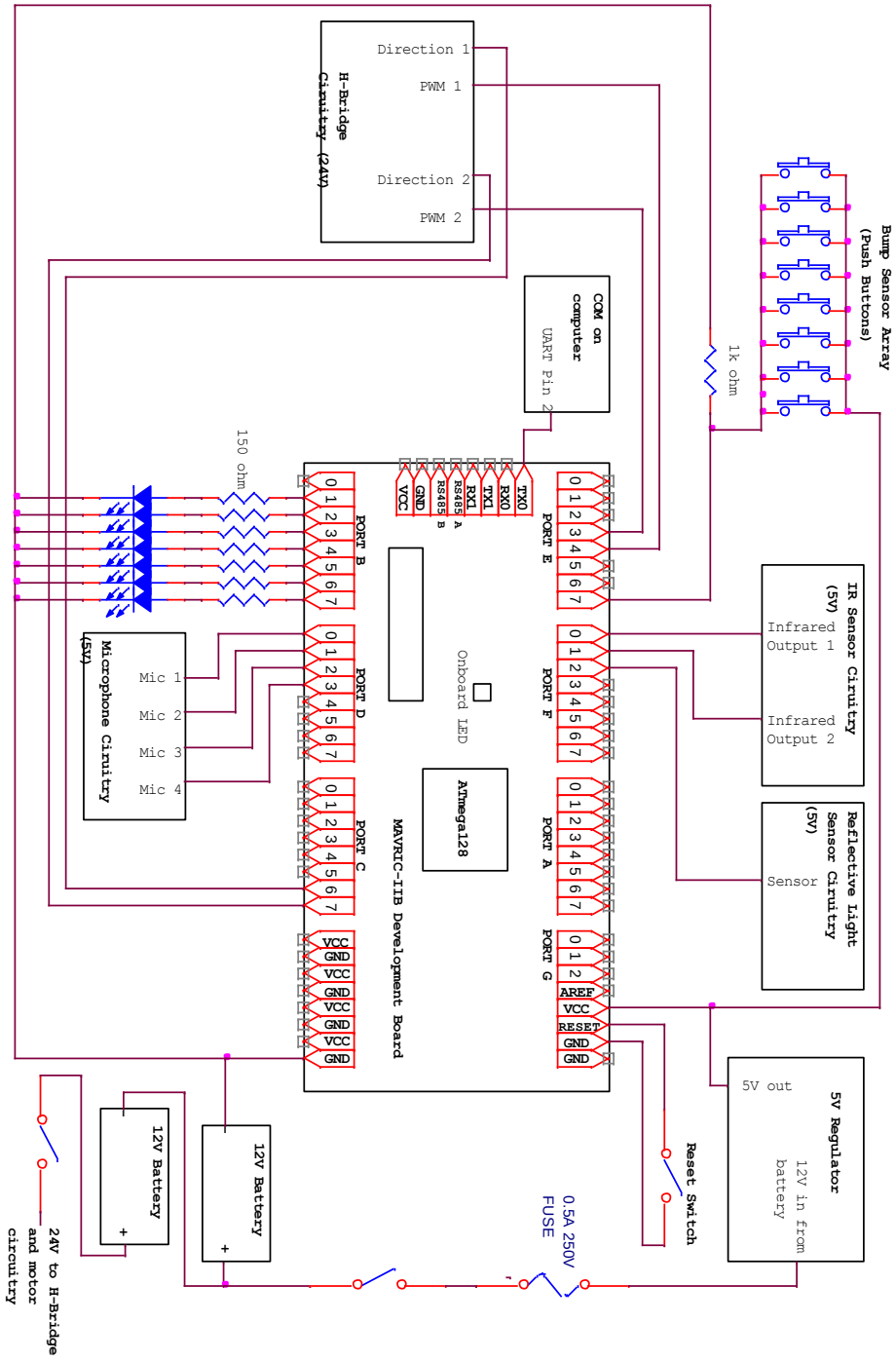



**Figure 14: Overall Circuit Diagram**

Figure 15 shows the circuit diagram of the LMD18200 H-Bridges connected to the motors and diodes. The diodes can be seen on the left side of the circuit diagrams and were implemented to cut out high voltage spikes. The LMD18200 H-Bridge is mounted on an Acroname EMF board which includes other external circuitry. With exception to external capacitors mounted on the EMF board, we are not using any other circuitry available. Each LMD18200 pin runs to an input, which is across the bottom of the board, and the output is on the right to the motors.

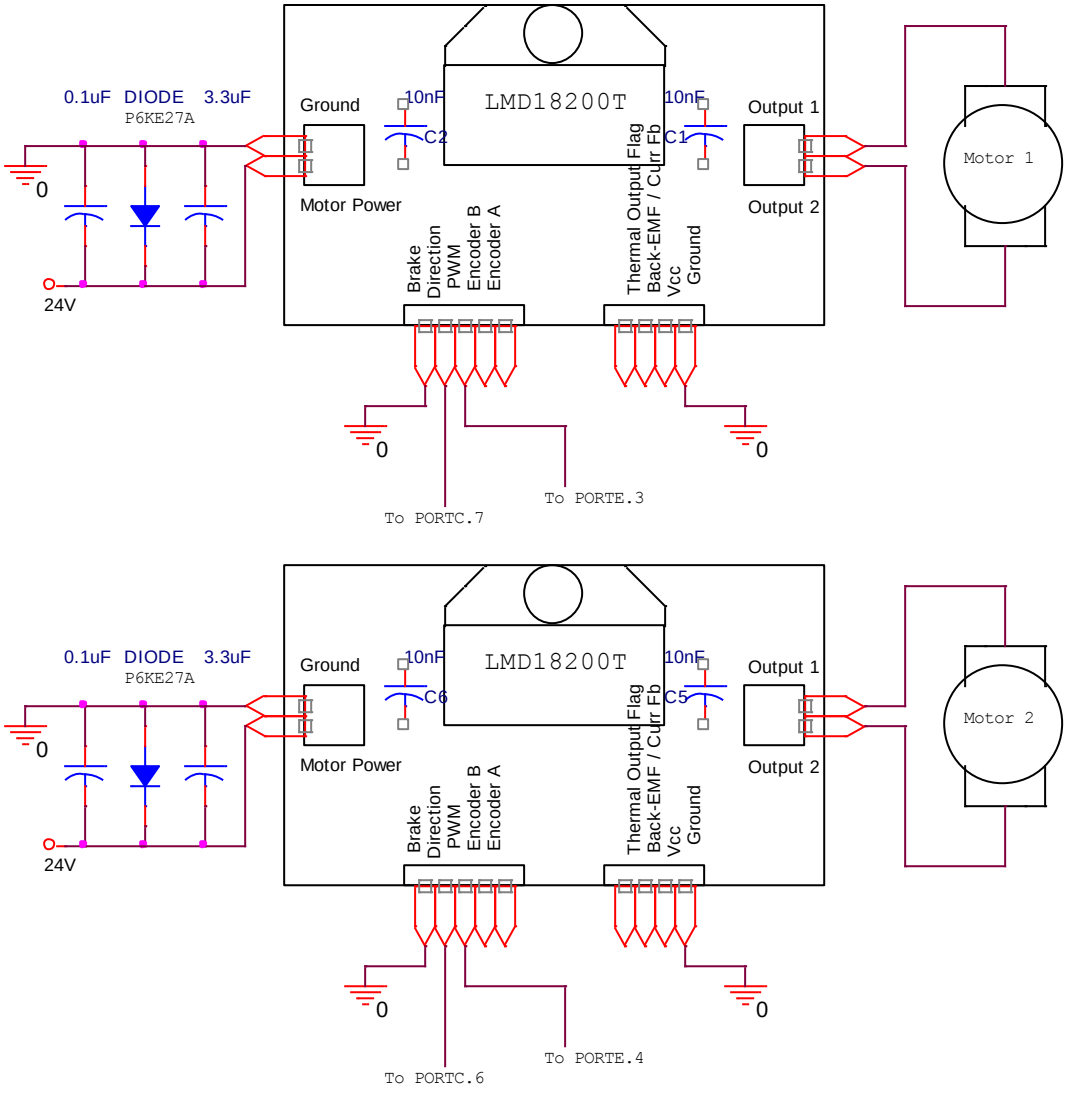



**Figure 15: Motor and H-Bridge Circuitry**

Figure 16 shows the microphone, amplifier, and low pass filter circuitry that is used for the project. The diagram only shows a single circuit but this is duplicated for four different microphones using the same LM 324A Quad Op Amp.

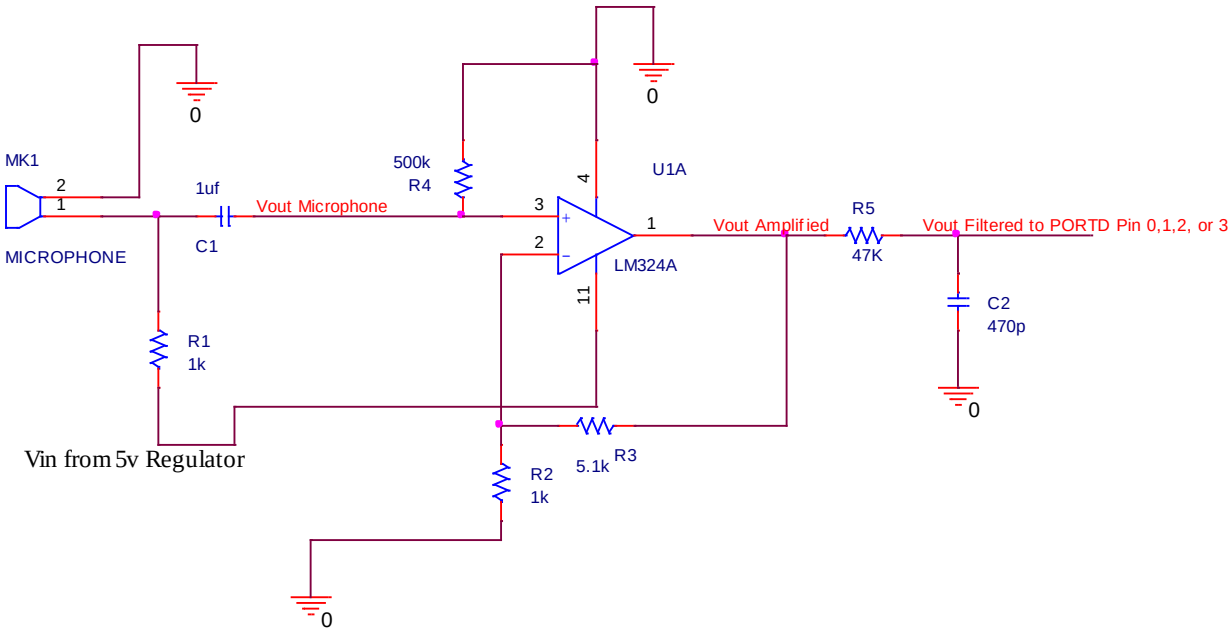



**Figure 16: Microphone, Amplifier, and LPF Circuitry**

Figure 17 shows the ultrasonic receiver circuitry that was never implemented onto EBR but was constructed and tested. This circuitry includes an amplifier to allow for an output that is readable on the ATmega128 microcontroller.

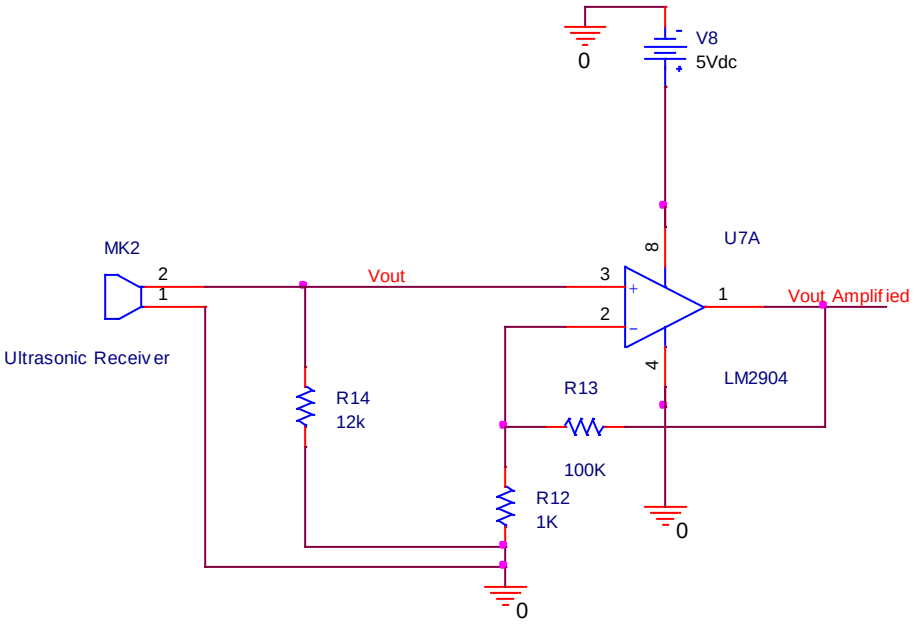



**Figure 17: Ultrasonic Receiver Circuitry**

Figure 18 displays the reflective light sensor circuitry. This circuitry includes the amplifier that is implemented to allow the ATmega128 to read the output from the reflective light sensor.

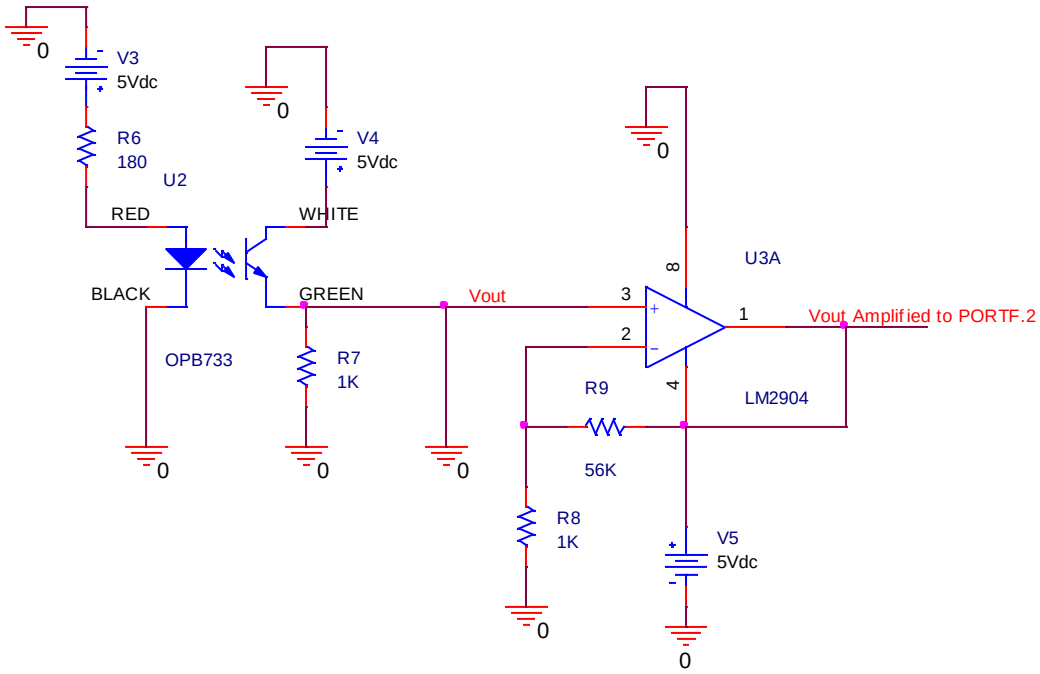



**Figure 18: Reflective Light Sensor Circuitry**

Figure 19 shows the voltage regulator circuit added to give the MAVERIC-IIB and other circuitry a clean 5volt output.

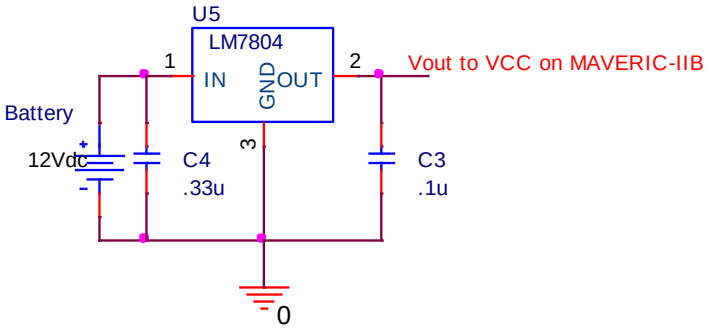



**Figure 19: Voltage Regulator Circuitry**

# *Appendix B: Design Equations*

Appendix B is a list of the calculations used during the design process of some of the hardware. Also included in Appendix B are the calculations for determining if the AtMega128 microcontroller is fast enough and determining the battery life for EBR.

### Calculation 1: Calculating R & C for Microphone Low Pass Filter

Calculation 1 is used for determine the Resistor and Capacitor values for the microphone low pass filter.

Fc=Cutoff Frequency of the Low Pass Filter.

The general equation for cutoff frequency can be seen below:

$$F_c = \frac{1}{2\pi RC} \tag{2}$$

Since more options are available for a resistor, solving Fc for the Resistor by using 470pF capacitor.

$$7000 = \frac{1}{2\pi R * 470 * 10^{-12}} \tag{3}$$

Solving for R gave 48375.4 Ω which is approximately 47000 Ω

### Calculation 2: Calculating Actual Cutoff Frequency of LPF in Microphone Circuitry

After calculating resistor and capacitor values, the cutoff frequency is recalculated in Calculation 2.

$$F_c = \frac{1}{2\pi RC} = \frac{1}{2\pi * 47000 * 470 * 10^{-12}} = 7204.84 Hz \tag{4}$$

### Calculation 3: Gain of Op Amps in Microphone Circuitry

Calculating gain of an op amp for the microphone circuitry is seen below. The General equation for gain is as follows:

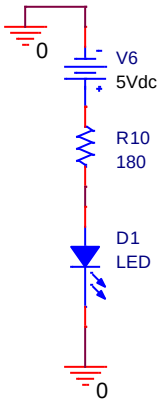
$$Gain = 1 + \frac{R2}{R1} \tag{5}$$

With: R1=1KΩ resistor R2=100KΩ resistor

$$Gain = 1 + \frac{100000}{1000} = 101 \tag{6}$$

### Calculation 4: Resistor Calculation for LED in Reflective Light Sensor

To ensure that the LED in the reflective light sensor operates at a safe current Calculation 4 is determining a safe resistor value.




Given values for current: $I_{MAX} = 50mA$ $I_{Safe} = 30mA$

General equation for calculating Resistor using voltage and current:

$$R = \frac{V}{I} \tag{7}$$

Using voltage and current values for reflective light sensor operation

$$R = \frac{5v}{30mA} = 166.66\Omega \tag{8}$$

Using R=180 Ω recalculate current

$$I = \frac{V}{R} = \frac{5v}{180\Omega} = 27mA \tag{9}$$

### Calculation 5: Calculating Speed Needed for Sound Detection

Calculation 5 is to determine if the microcontroller is fast enough to detect the difference between two microphones distanced 8 inches apart.

Distance between microphones = 8 inches

Speed of sound = 340.29 m/s = 34029 cm/s = 13,397.2441 inches/sec

$$Clockspeed needed = \frac{Speed of sound}{Dis\_between Mics} = \frac{13{,}397.2441}{8} = 1.6747 kHz \quad (10)$$

The speed required was found to be 1.6747 kHz and the ATmega128 runs at 16 MHz.

### Calculation 6: Calculating Battery Life of EBR

Calculation 6 is to determine the battery life of EBR using the current draw of all the components and the current supply of the batteries being utilized.

| Component | Volts | Amp |
|---|---|---|
| Motors & H-Bridges | 26.09 | .596 |
| Electronics | 5.116 | .178 |
| Voltage Regulator | 7.634 | .178 |
| Batteries | 12 | 2.3AmpHours |

$$Total Current = (\frac{.596}{2} + .178 + .178) = .654 Amps \quad (11)$$

$$Battery Life = (\frac{2.3}{.654} * .8) = 2.8135 Hours \quad (12)$$

# *Appendix C: Equipment*

Table 3 contains all the parts necessary to complete the project. The table does not include the cost of items available in the lab, such as wire, resistors, capacitors, and other circuitry items. It also does not include the price of materials used to build the chassis.

**Table 3: Parts List**

| Component | Part Number | Quantity | Crucial Spec | Unit Cost | Ordering Cost |
|---|---|---|---|---|---|
| MAVRIC-IIb | MAV2BPH16 | 1 | | $ 99.00 | $ 99.00 |
| Motor | 0-BHG31 | 2 | Torque and RPM | $ 23.99 | $ 47.98 |
| Wheels | 0-DAV5540 | 2 | Diameter | $ 8.99 | $ 17.98 |
| Hub | 0-MHUB04 | 1 | | $ 8.49 | $ 8.49 |
| H-Bridge | Use H-Bridge from Mini Project | | Max Current | | |
| Microphone | CMB-6544PF | 5 | | $ 0.72 | $ 3.60 |
| IR Sensor | Sharp GP2Y0A21YK | 4 | Distance(Min & Max) | $ 12.50 | $ 50.00 |
| Reflective Light Sensor | 365-1510-1-ND | 2 | Reading Distance | $ 2.90 | $ 5.80 |
| Push Button | 149948 | 8 | Momentary Switch | $ 0.29 | $ 2.32 |
| | | | | | |
| | Total | | | | $ 235.17 |

# *References*


[1] N. Nilsson, “Artificial Intelligence: A New Synthesis”, San Francisco, CA: Morgan Kaufmann, 1998.

[2] R. Cioarga, B. Ciubotaru, D. Chiciudean, M. Micea, V. Cretu, and V. Groza, “Emergent Behavioral Modeling Language in Obstacle Avoidance”, Warsaw, Poland, May 2007.

[3] Galen Carol Audio, "Decibel (Loudness) Comparison Chart," [Online document], 2007, [cited 2009 Nov 11], Available HTTP: http://www.gcaudio.com/resources/howtos/loudness.html