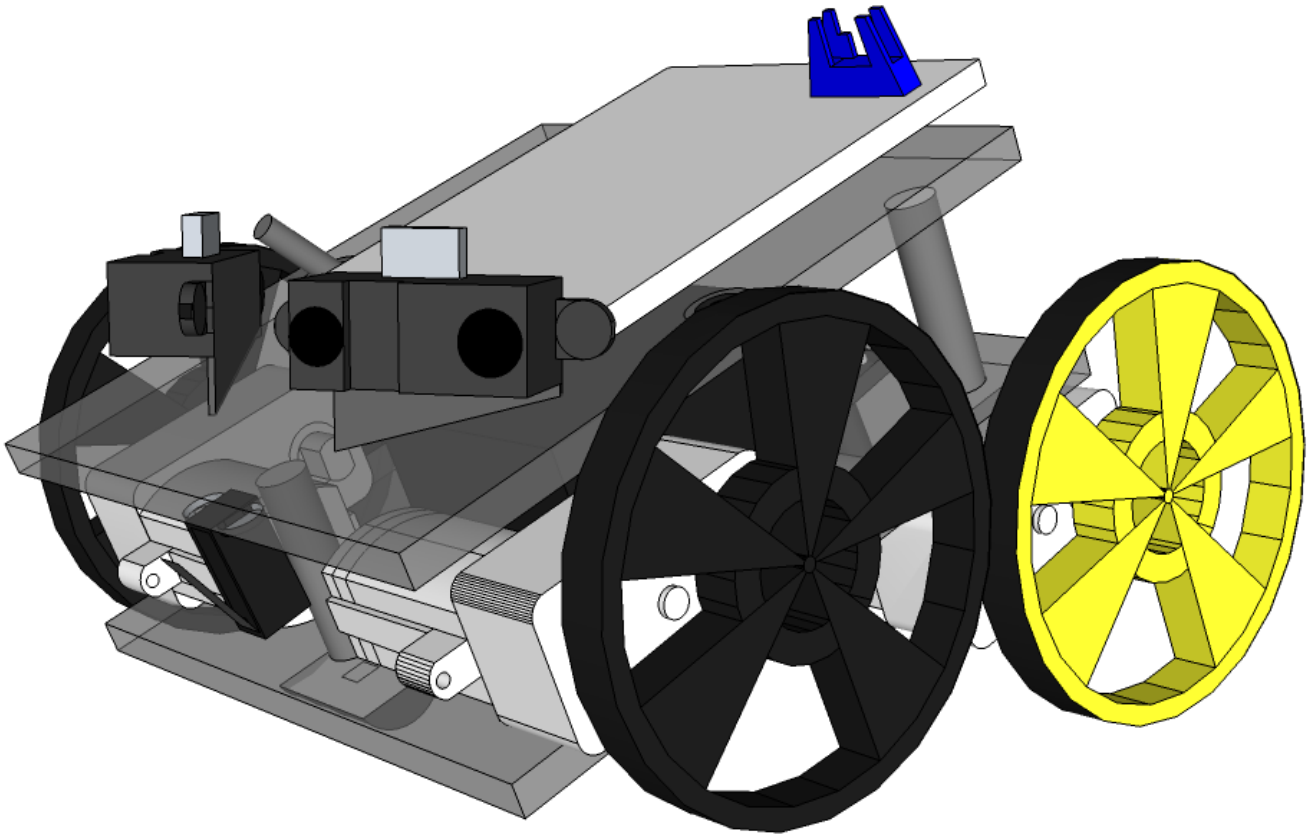
SUMO BOT



Technical Manual

Prepared By:

TEAM S.U.M.O.B.O.T.

Table of Contents

1 Target Audience

This Technical Manual is intended to be used by the following audiences:

- robotics enthusiasts
- computer & electrical engineers
- technicians, technologists
- electronics repair persons

2 Statement of Purpose

The goal of this Technical Manual is to present you, the reader, with the design and implementation of this SUMO BOT product in a way that will allow you to become thoroughly familiar with both the hardware and the software of the SUMO BOT. You should be able to, in fact, fully disassemble the robot, and then put it all back together using this Technical Manual as a guide.

3 Hardware Components

Hardware comprising the SUMO BOT can be broken down into these top categories :

- chassis
- sensors
- motors
- wheels
- ball caster
- brainboard (PCB with soldered electronic components)
- mounting brackets + metal screws
- connecting wires
- jack socket screws
- battery pack

We are using two transparent plexiglass **chassis** of these dimensions: width = 106 mm, length = 112 mm, thickness = 5 mm. Each has predrilled, threaded holes capable of accomodating M3-0.5 screws.

Mounted on upper chassis, there are two Sharp GP2Y0A02YK range IR **sensors.** Mounted on lower chassis, facing the ground, there is one OPB606A opto-reflector IR edge **sensor.** Mounted on the upper chassis, there is one lever-actuated contact switch.

All four **motors** are Solarbotics GM8 DC motors. The front-left and front-right units are mounted to the top surface of the bottom chassis. The rear-left and rear-right units are mounted to the bottom surface of the bottom chassis. This is what gives the robot its slanted hot-rod look.

All four **wheels** are Solarbotics GMPW wheels [diameter = 65 mm, width = 7.62 mm]. These wheels have silicone tires. They tightly fit onto the shafts of GM8 motors, holding onto the axles through friction.

A single **ball caster** (by Tamiya) sits mounted to the backside of the bottom chassis, between the two rear wheels. It is being used to provide a third, rear point of contact for our robot.

The **brainboard** is a PCB that ships with all electronic components already soldered on. The two components that are easily removable without resorting to a soldering iron are the ATmega8L microcontroller chip, and the SN754410N quad half-H bridge chip; both are seated in their respective sockets. The PCB features headers for sensors, motors, and an ISP (In-System Programming) cable. Sensors and motors use 3-pin molex connectors. The ISP cable attaches using a custom 10-pin connector. The **brainboard** attaches to the upper chassis using 4 M3-0.5 screws.

Each range IR sensor is mounted on a metal L-shaped **bracket**. One point of attachment is the sensor mounting hole, the other joint location is a threaded hole in the upper chassis. The ground-facing opto-reflector IR edge sensor uses its custom, collapsed-U metal **bracket** that is attached to the bottom chassis. The edge sensor is mounted on a PCB wafer that slides into that bracket. The four motors use a custom-shaped metal **bracket** that hugs the molded plastic housing of each motor. Each bracket is attached to the bottom chassis using two mounting (threaded) holes. Each bracket attaches to the motor that it braces using one screw.

As previously mentioned, **connecting wires** are used to create electrical connections between the PCB and these hardware components: sensors, motors, ISP connector. For the range IR sensors, both extremities of **connecting wires** end in 3-pin molex connectors. At the edge IR sensor, the wires are soldered on, while the other end has a molex connector. At the motor housing end, the wires are soldered on, while the other end has a molex connector. For the ISP cable, one end has a 10-pin connector while other other extremity ends in a soldered on USB connector. The leads coming from the power pack use a custom power connector.

We used columns of three **jack-socket** screws fastened into each other, as well as into threaded holes in bottom and top base chassis to create a double-decker structure. Two such columns are in the frontal part of the robot, one column supports the chassis in the rear part of the robot.

The **battery pack** consists of 6 x AA NiMH cells connected in series, providing a total of 2500 mA/h. It is attached to the center of the underside of the top chassis, using a metal U-shaped bracket with 2 screws, which enter the threaded chassis holes from the bottom of the chassis.

In the following table, you'll find all electronic components found on the **brainboard** PCB.

Schematic Symbol	Part Name	Description	Amount Needed	Ratings
U1	LM7805CT	Voltage Regulator	1	+5 V, 1 A
U2	ATmega8L	8-Bit Microcontroller	1	2.7 V to 5.5 V
U7	SN754410	Quad Half-H Bridge	1	36 V
CN1	DIP 10 PIN	Serial Transfer Cable Connector	1	-
C1	0.33 µF	Capacitor	1	50 V
C2,C3,C4,C5,C6	0.10 µF	Capacitor	5	50 V
R1	10 kΩ	Resistor	1	¼ Watt
R2	330 Ω	Resistor	1	¼ Watt
R3,R5	150 Ω	Resistor	2	¼ Watt
R4,R6	6.8 kΩ	Resistor	2	¼ Watt
LED1	LED	Red LED	1	-
S2	Reset Button	Reset Push-Button	1	-
Vs	Battery	9 V Battery	1	-

Table 3.1: Electronic Components on the Brainboard PCB

In the following table, you'll find all hardware components (considered to be an integral part of the robot) found **outside** of the brainboard PCB:

Schematic Symbol	Part Name	Description	Amount Needed	Ratings
Vc	Battery Pack	6xAA NiMH Battery Pack	1	9V, 2500 mA/h
S1	Toggle Switch	Lever-Actuated Toggle Switch	1	-
U5,U6	OPB606A IR Sensor	Opto-Reflector IR Edge Sensor	2	-
U3,U4	Sharp GP2Y0A02YK IR Sensor	IR Range Sensor	2	-
S3	Contact Switch	Contact Switch	1	-
M1,M2,M3,M4	Solarbotics GM8 Motor	Gear Motors	4	-
-	Range Sensor Bracket	Range Sensor Bracket	2	-
-	Edge Sensor Bracket	Edge Sensor Bracket	1	-
-	Motor Bracket	Motor Bracket	4	-
-	Solarbotics GMPW Wheel	Plastic Wheel, Silicone Tires	4	-
-	Ball Caster	Tamiya Ball Caster	1	-
-	Base Platform	Plexiglass Chassis	2	-
-	M3-0.5 Screws	Metal Screws	21	-
-	Connecting Wires	Range Sensor to PCB Connection	2	-

Table 3.2: Integral Hardware Components of the Robot, Found Outside of the Brainboard PCB.

4 Block Diagram of the Sumo Bot

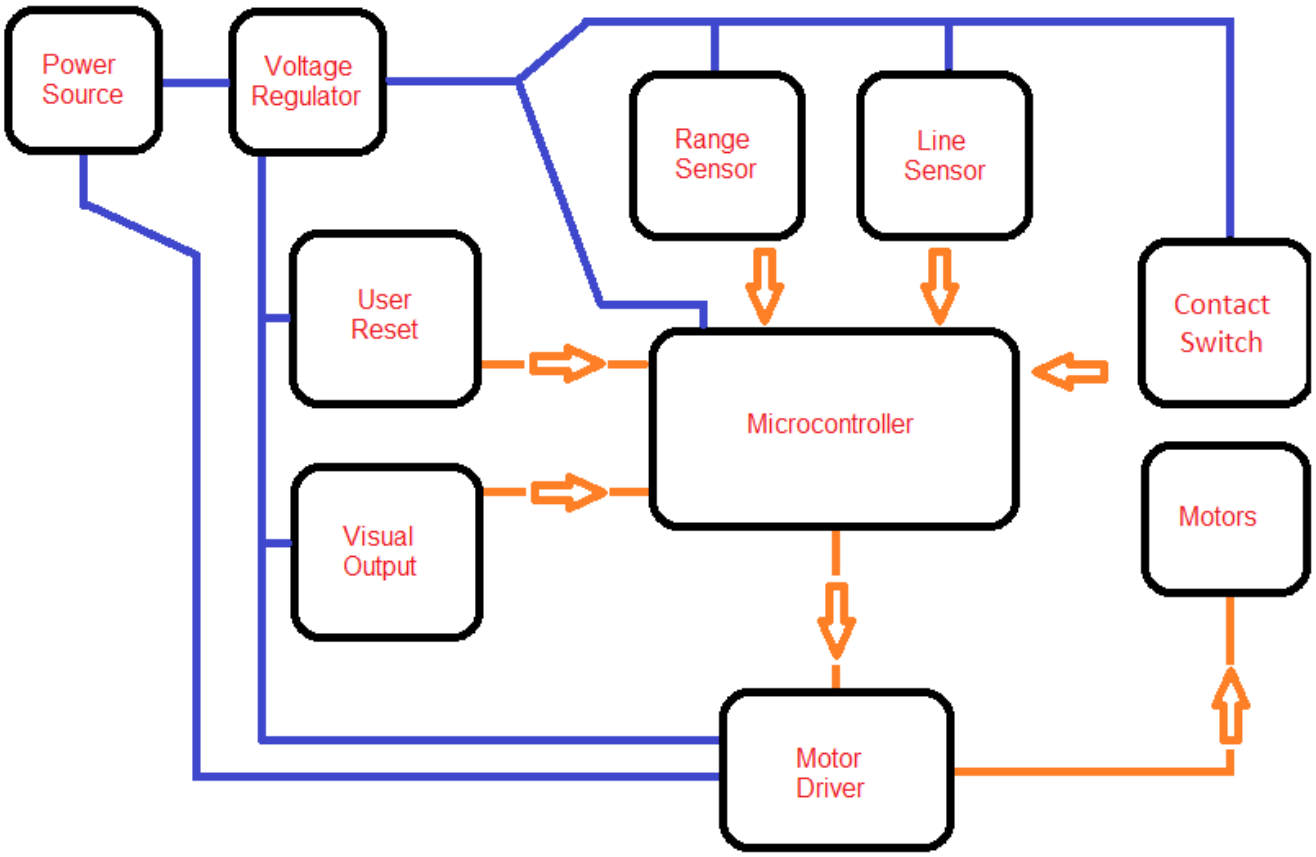


Figure 4.1: Block Diagram of the Sumo Bot

5 Circuit Schematic of the Sumo Bot

The following is the schematic of the SUMO BOT, showing **both** the PCB components as well as sensors, motors, battery pack, and toggle switch:

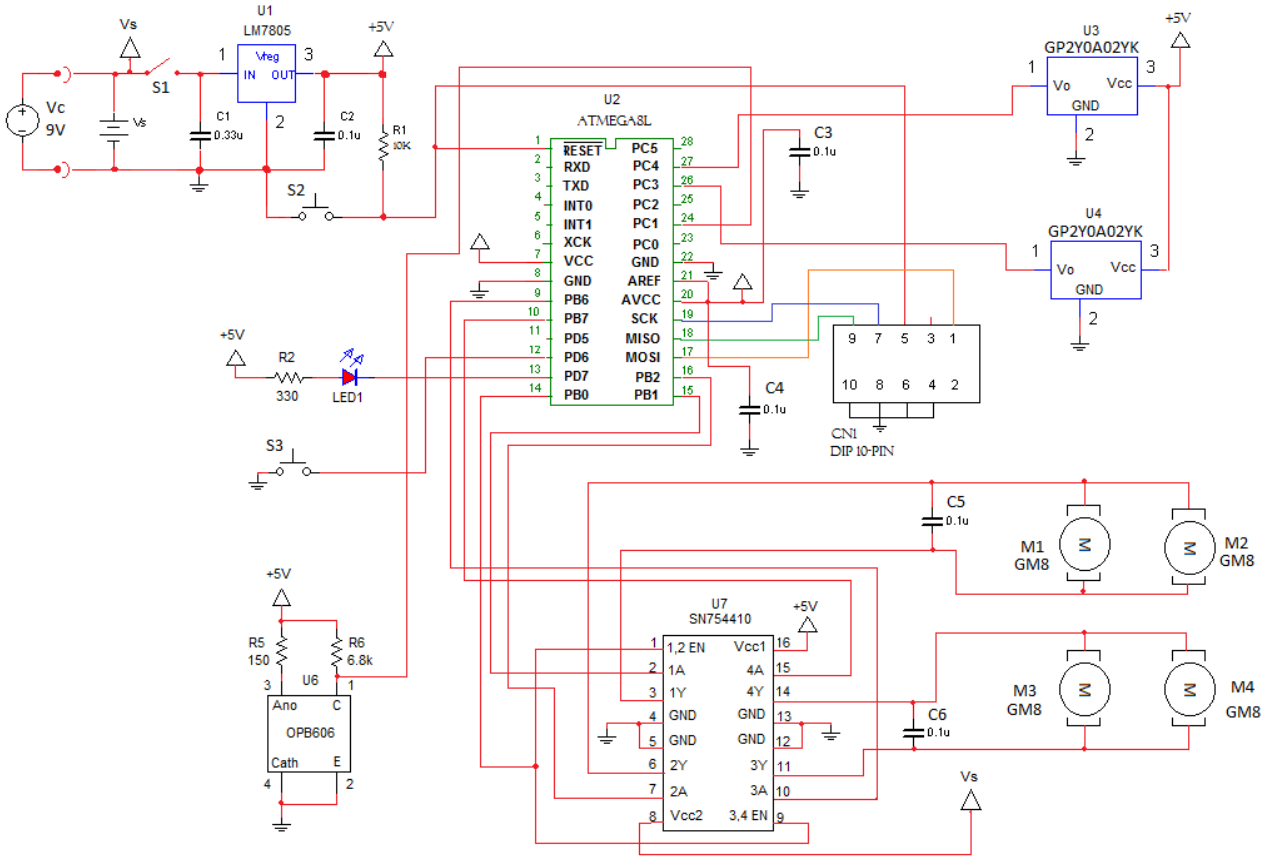


Figure 5.1 Circuit Schematic of the Sumo Bot

6 How Electronic Components Integrate With Each Other

6.1 Voltage Regulator – LM7805 (+5 V DC regulator)

The voltage regulator reduces the input voltage and regulates it to obtain a constant voltage output. We connect a battery source that is more than +5 V DC to the input terminal, and connect the common terminal (ground) to the negative battery terminal. Pin 3 is our output voltage which will be a constant 5 V providing that the input does not drop below 7 V.

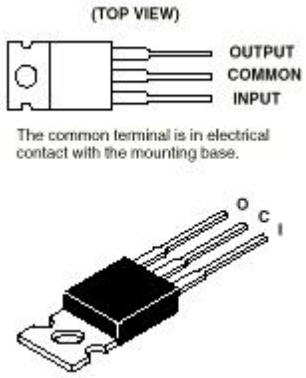


Figure 6.1: LM7805 +5 V DC Regulator[1]

We connect coupling capacitors to the input and output to ground as seen in Figure 6.2 below.

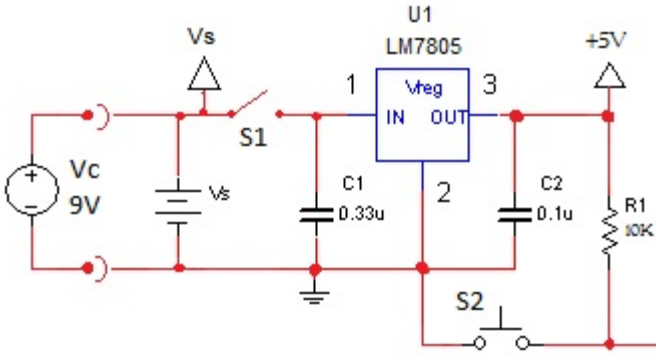


Figure 6.2: 5 V Power Supply Circuit

6.2 ATmega8L Microcontroller

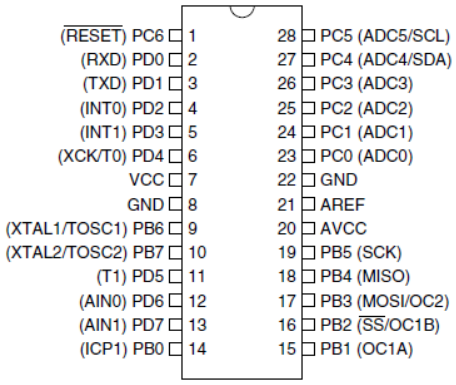


Figure 6.3: Pin Functions of Atmel ATmega8L Microcontroller

The microcontroller is the "brain" of the project. We are using the AVR 8-bit ATmega8L microcontroller that provides us with three input/output ports. This can include up to 6 analog to digital converters, two hardware interrupts along with several different functions. We will be using the hardware interrupts (pin 4 and 5, INT0 and INT1) for line sensing, and ADC0 and ADC1 (pins 23 and 24) for the analog range sensors. The wheels will be controlled by the H-bridges connected to pins of port B. (Please see Figure 6.4)

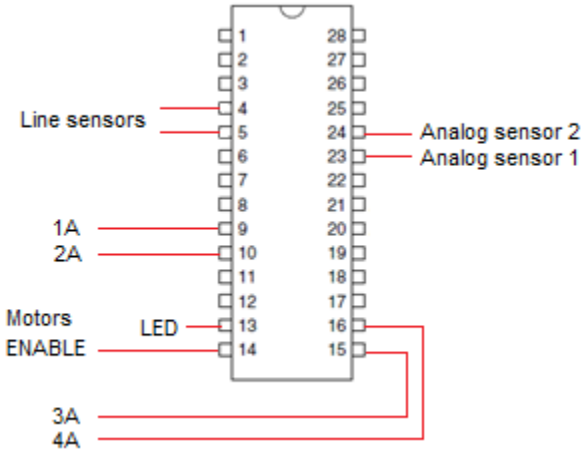


Figure 6.4: Connections to exterior components

To power this chip, we will also connect pins 7 and 20 to +5 V from the voltage regulator, and pins 8 and 22 to ground. In order to reset the chip we must force pin 1 to go to an active low state. To do this we attach a pull up resistor between +5 V and pin 1, then pin 1 to a normally-open switch, and then the second terminal of the switch to ground. Pressing the switch and releasing it results in the microcontroller resetting.

For analog to digital conversion, which is necessary in this case to read the analog sensors, a reference voltage is necessary (V_{REF}). This will be done internally through a program-selectable voltage regulator providing 2.54 V to the ADC. Since this is the case, no external voltage reference is necessary.

To program the chip control, signals are passed to it through MISO, MOSI, SCK and RESET using ISP connection as seen in the circuit diagram. Seeing as this is only used during programming, the final robot will not have this feature on it. Figure 6 shows the connections necessary to allow communications between the computer and the ATmega8L microcontroller.

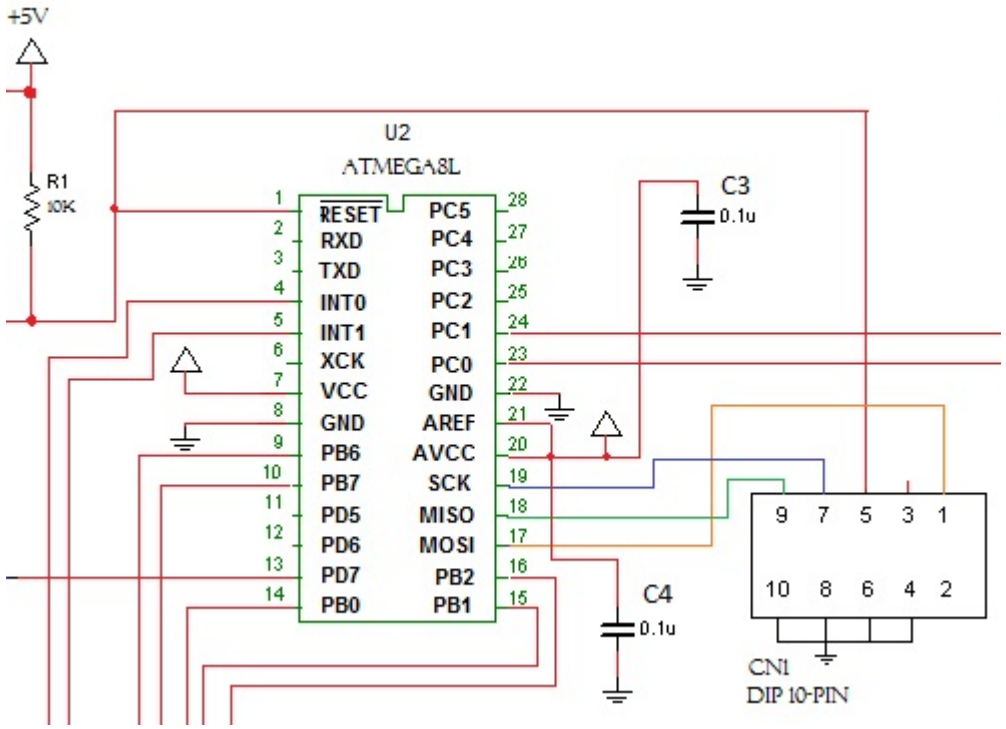


Figure 6.5: ISP connections for programming the microcontroller

6.3 Motor Driver: SN754410N (Quad Half-H Bridge)

The motor driver is a key element of integrating the motors with the microprocessor. Normally, when voltage is applied to one lead of a DC motor and ground is connected to the other lead, it causes a DC motor to turn, and if we switch the polarity of the leads, the motor will go in reverse. There are several major shortcomings with this setup, given that the microcontroller cannot provide enough current or voltage to power the motor effectively. Also there is no easy way to make the motor change directions. This is where the "H-Bridge" is used; it allows for a secondary voltage source to be used but also allows for forward and backwards movement.

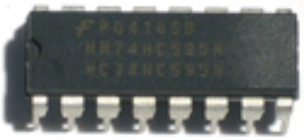


Figure 6.6: SN754410N Quad Half-H Bridge Package[2]

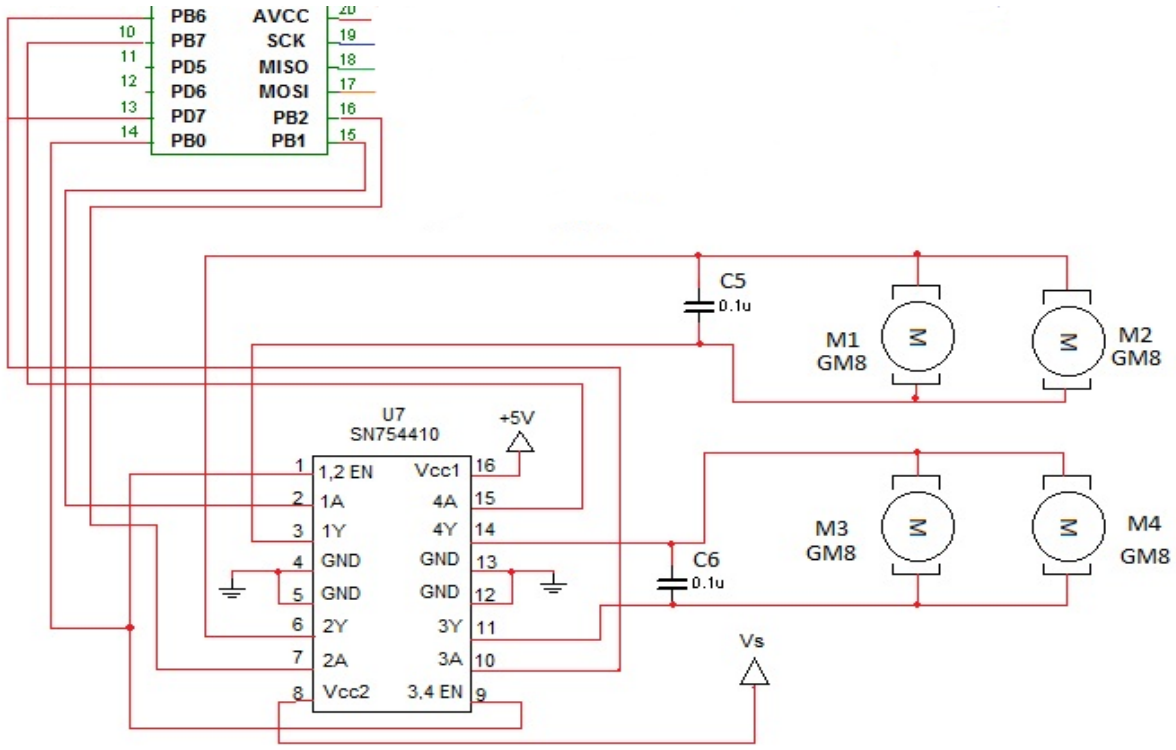


Figure 6.7: H-Bridge Chip Connections

The H-Bridge has two voltage sources (see Figure 6.7), Vcc1 for powering the chip and Vcc2 for powering the motors. There are four connections for ground (GND) which can also be connected to a heat sink, if needed. The chip allows for a maximum current draw of 1.1 A, so it is important to keep this in mind. The inputs 1A, 2A, 3A and 4A are connected to output ports on the microcontroller and can control the outputs 1Y, 2Y, 3Y, and 4Y directly if the enable is activated. There are two enable pins on the chip which are active high, so providing them with a +5V signal will activate the output pins. We have both 1,2EN (1 and 2 output enable) and 3,4EN (3 and 4 output enable) attached together in order to turn on all outputs with a single pin. Output pins 1Y and 2Y are connected to the left-side motors and output pins 3Y and 4Y are connected to the right-side motors. This will allow for forward and backward movement by each of the motors.

6.4 Solarbotics GM8 Motors (Gear Motor 8 Offset Shaft)

The motors are the parts that make the robot move. By applying voltage to one lead and grounding the other, we can get the DC motor to turn clockwise, and if we reverse these leads we can get the motor to turn counter-clockwise.

They are held in place by a mounting bracket fastened to the side of the body (see Figure 6.8). The mounting brackets allow for the motor to spin freely while moving the body with it.

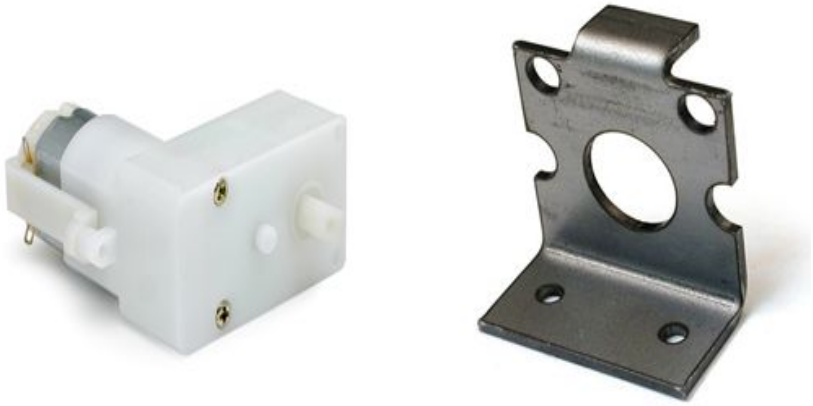


Figure 6.8: GM8 Motor[3] and Mounting Bracket[4]

Once the motors are secured to the mounting hardware, the mounting bracket it attached to the chassis of the robot.

Technical Information (obtained from Solarbotics GM8 datasheet):

	3V	6V	Value
Gear Ratio	--	--	143:1
Unloaded RPM	40	78	--
Unloaded Current	50mA	52mA	--
Stall Current	400mA	700mA	--
Stall Torque	44.44 in*oz	76.38 in*oz	--
Length	--	--	55mm
Width	--	--	48mm
Height	--	--	23mm
Weight	--	--	32.00 g

6.5 GMPW Plastic Wheels

The wheels provide the motors with movement capabilities. The diameter of these wheels is 69 mm and they are 7.62 mm wide. They are moulded plastic with silicone tires, providing the robot with good traction. As seen in Figure 6.12 below, these wheels fit directly to the GM8 motors used in this project.

Figure 6.9: GMPW Plastic Wheels[5]

6.6 Sharp GP2Y0A02YK Analog Range Sensors

These range sensors use infrared light to measure the distance between the lens and objects in front of it. This particular model has analog output, which means that the output is a voltage that varies between 0.4 V and 2.4 V depending on how far or how close an object is. It has a range of approximately 20 cm to 150 cm. Since the output is a voltage, we will have to pass it through our Analog to Digital converter (ADC) in order to obtain a numerical (binary) value.

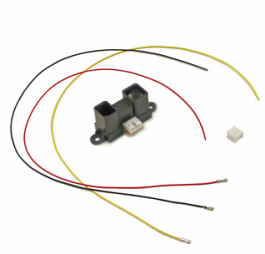


Figure 6.10: Sharp Analog IR Range Sensor, GP2Y0A02YK[6]

To connect the sensors to out circuit, we attach pin 1 of the sensor to the Analog to Digital converter (ADC). Pin 3 is connected to +5 V and pin 2 is connected to ground. To identify pin 1 on the sensor, place the pins facing you, and point the lenses upward. Pin 1 is the left most of the 3 pins.

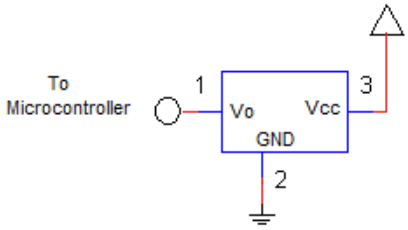


Figure 6.11: Range Sensor Wiring Diagram

When positioning the sensors, we will need to make sure to adjust the angle of the sensors to ensure that they are not angled towards the floor. To do this, ensure that the both lenses of the sensor are facing parallel to the floor; this will ensure that the readings will be more accurate.

6.7 OPTEK OPB606A Line Sensors

To keep the SUMOBOT inside the ring, a system of line sensors ensure that the robot knows when not to go any further. This is done by measuring the reflectivity of the floor, and knowing that black tape marks the edges, we can assume that when we hit something non reflective we are at the limits of the ring. To do this we are using the OPB606A sensor, which contains a light emitting diode (LED) and a photo-transistor receiver (see Figure 6.12).

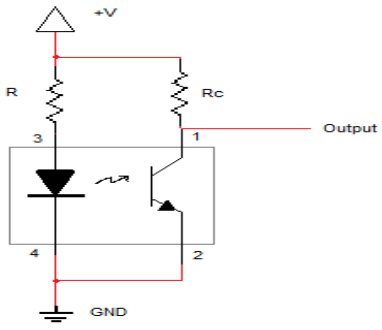


Figure 6.12: Line Detector Circuit

We will position the line detectors approximately 1 mm off the floor to ensure proper readings. When the light emitted from the LED is detected by the photo-transistor then the transistor is on and the output voltage is low. When a line is present there will be little light reflected and the transistor will be off, and the output voltage will be logic high (see Figure 5.16). The current passing through the LED is approximately 20 mA, and using a +5 V source we are using the resistance R as 150 Ω, assuming a voltage drop of 1.7 V across the LED. The transistor has a V_{CE} at saturation of 0.4 V and current of 0.6 mA, so the R_C value used will be 6.8 kΩ.

Figure 6.13: (left) Output voltage when away from black tape line, (right) Output voltage when over black tape line.

The output of these two line sensors will be provided to the microcontroller, one of the two sensors to pin 4 and the other to pin 5. These pins are interrupts which will cause an interrupt subroutine to take effect, thus alerting the microcontroller to the current position of the robot in relation to the edge of the circle. When this happens, the robot will execute a tactic to turn around without going out of bounds.

6.8 Contact Switch

A lever-actioned contact switch allows us to confirm frontal contact with the opponent robot. We can then signal to our robot to move forward at full speed, trying to push the robot out of the battle arena.

6.9 Tamiya Ball Caster

A ball caster is an addition to our sumo robot's design that allows the 2 front wheels of the robot to be outside of the ring, while more than 50% of its supporting parts (2 rear wheels + the ball caster) are within the ring.

Figure 6.14: Tamiya Ball Caster[7]

7 Assembly Instructions

Note: All screws used in this assembly process are metal screws that have a flat surface at right angles to the shaft, under the head, with threads running all the way from the end of the shaft to the screw head. All screws used in process are being used to create a tight fit between two components that need to be joined together. It is recommended that you use #2 size Phillips screwdriver tip.

Step 1: Attach GM8 motors.

(a) attach front-left motor to the top surface of the bottom chassis
(b) attach front-right motor to the top surface of the bottom chassis
(c) attach rear-left motor to the bottom surface of the bottom chassis
(d) attach rear-right motor to the bottom surface of the bottom chassis

Each motor is fastened to the chassis using an L-shaped metal bracket.

The brackets are joined to motors using 1 screw. The brackets are attached to the chassis using 2 screws. The chassis has pre-drilled, threaded holes, so we can use screws instead of nuts & bolts.

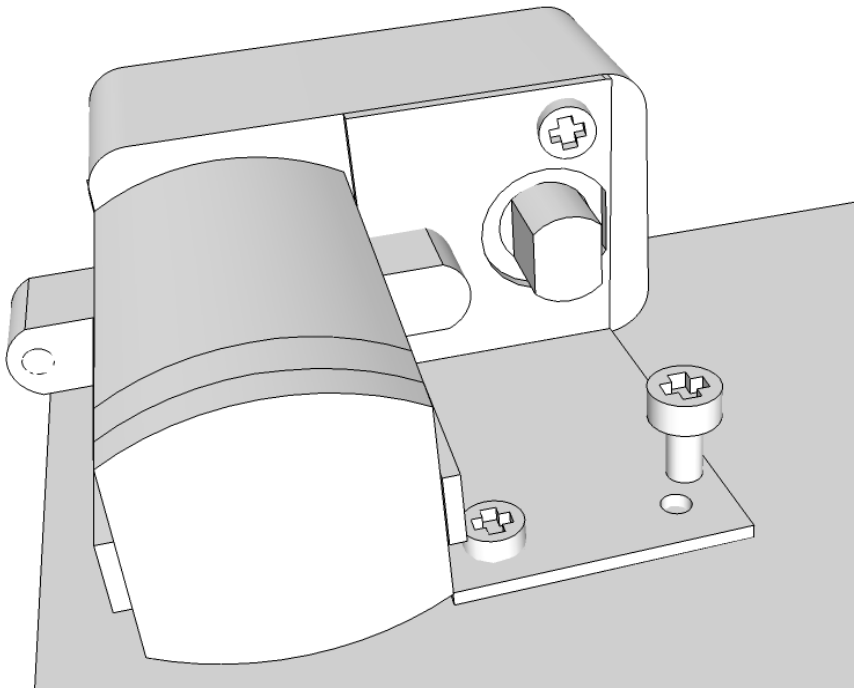


Figure 7.1: Fastening GM8 Motors to the Bottom Chassis

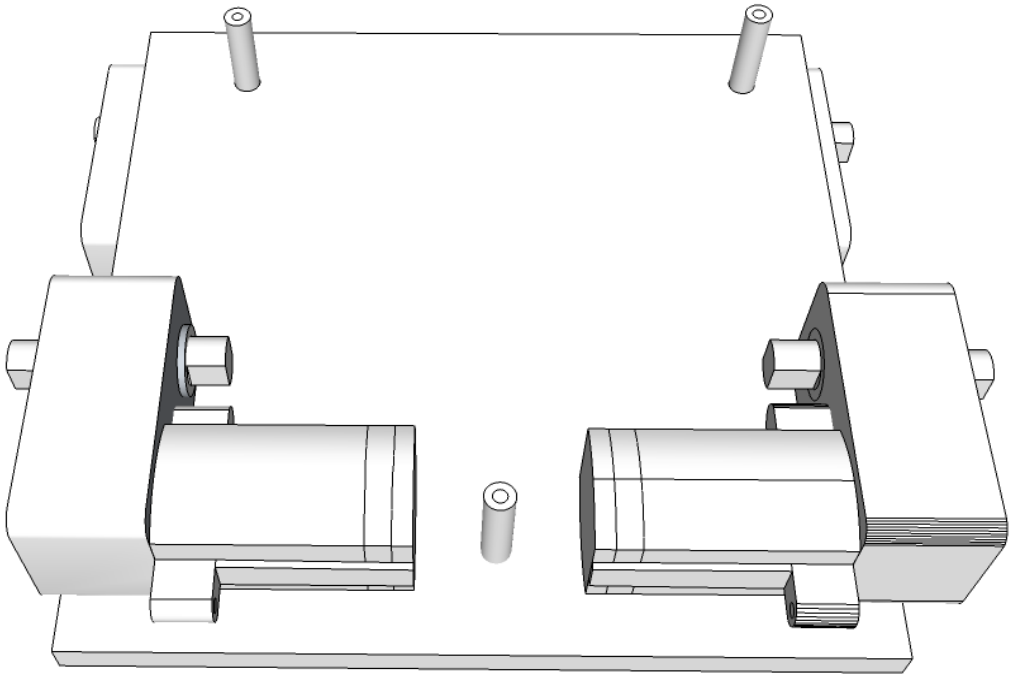


Figure 7.2: Visual of all 4 GM8 Motors Attached to the Bottom Chassis, Top View

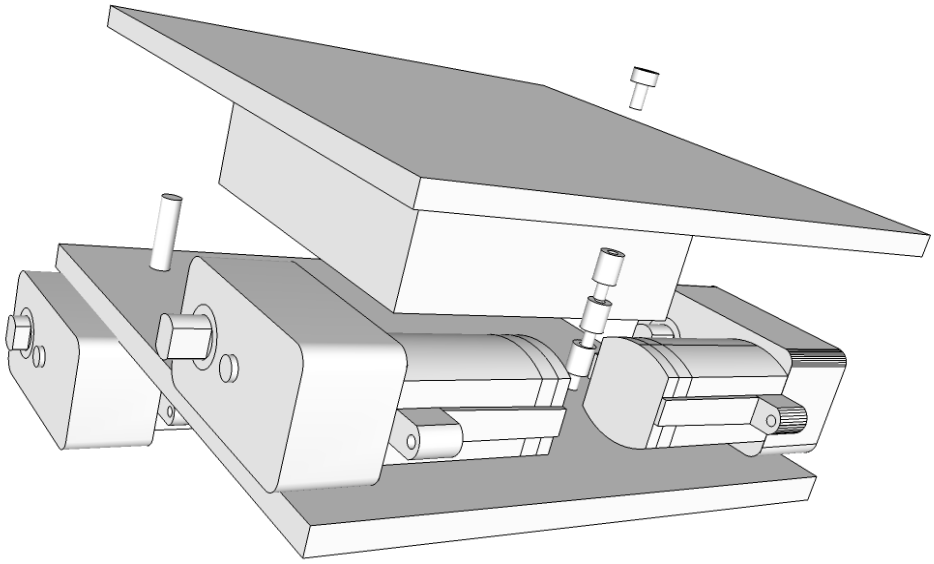


Figure 7.3: Visual of all 4 GM8 Motors Attached, Side-Diagonal View. Also Visible: Battery Pack, Top Chassis, Jack-Socket Screws

Step 2: Attach battery pack, on/off switch, charger connector.

Attach the battery pack to the center of the underside of the top chassis using a metal U-shaped bracket and 2 screws, which enter the threaded chassis holes from the bottom of the chassis. Fasten the on/off switch by pushing its threaded top through a pre-cut hole in the right center position of the top chassis and by tightening a hexagonal nuts until it is securely attached. Similarly fasten the charger connector.

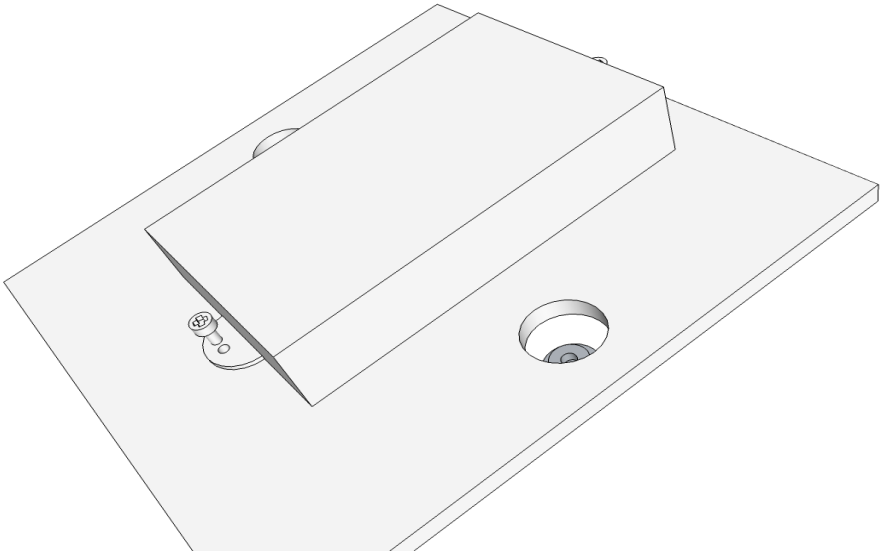


Figure 7.4: Battery Pack Being Attached to Top Chassis.

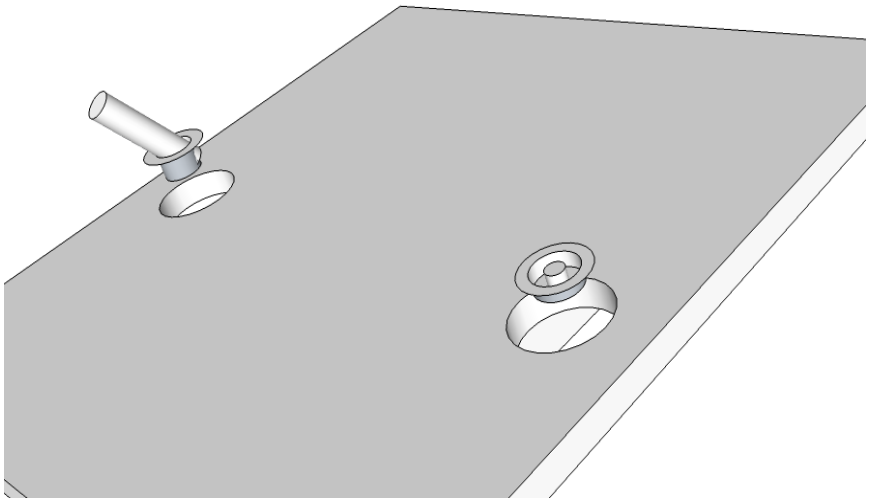


Figure 7.5: On/Off Toggle Switch (Left), Charger Connector (Right), Being Attached

Step 3: Form a double-decker structure consisting of bottom and top base platforms (chassis).

Use three columns of jack socket screws (fastened into each other), as well as into threaded holes in bottom and top base chassis to create a double-decker structure. Two such columns are in the frontal part of the robot, one column supports the chassis in the rear part of the robot.

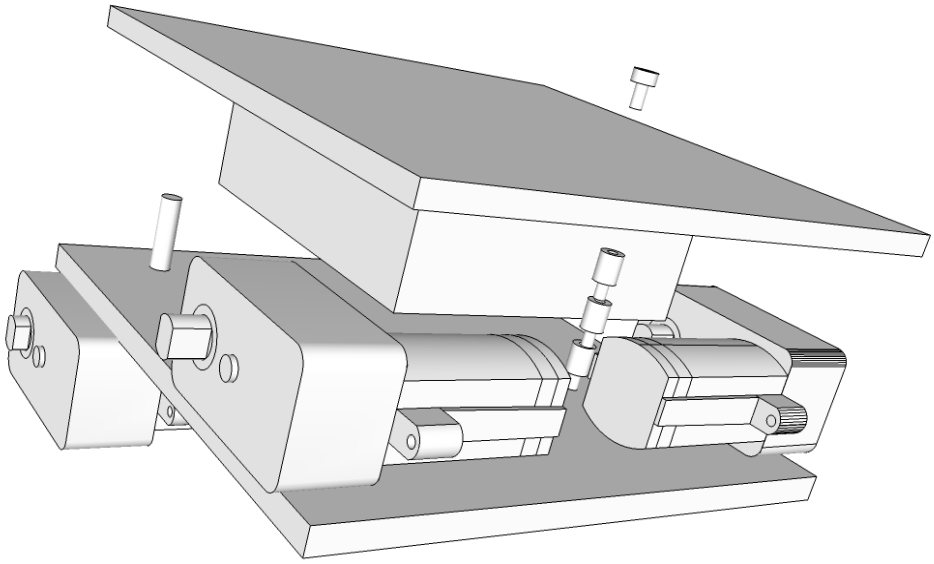


Figure 7.6: Visual of How the Double-Decker Structure is Constructed Using 2 Plexiglass Chassis & 3 sets of 3 Jack-Socket Screws. The Battery Pack is Mounted Under the Top Chassis.

Step 4: Attach PCB brainboard to the upper chassis.

Attach PCB brainboard to the upper chassis using 4 M3-0.5 screws.

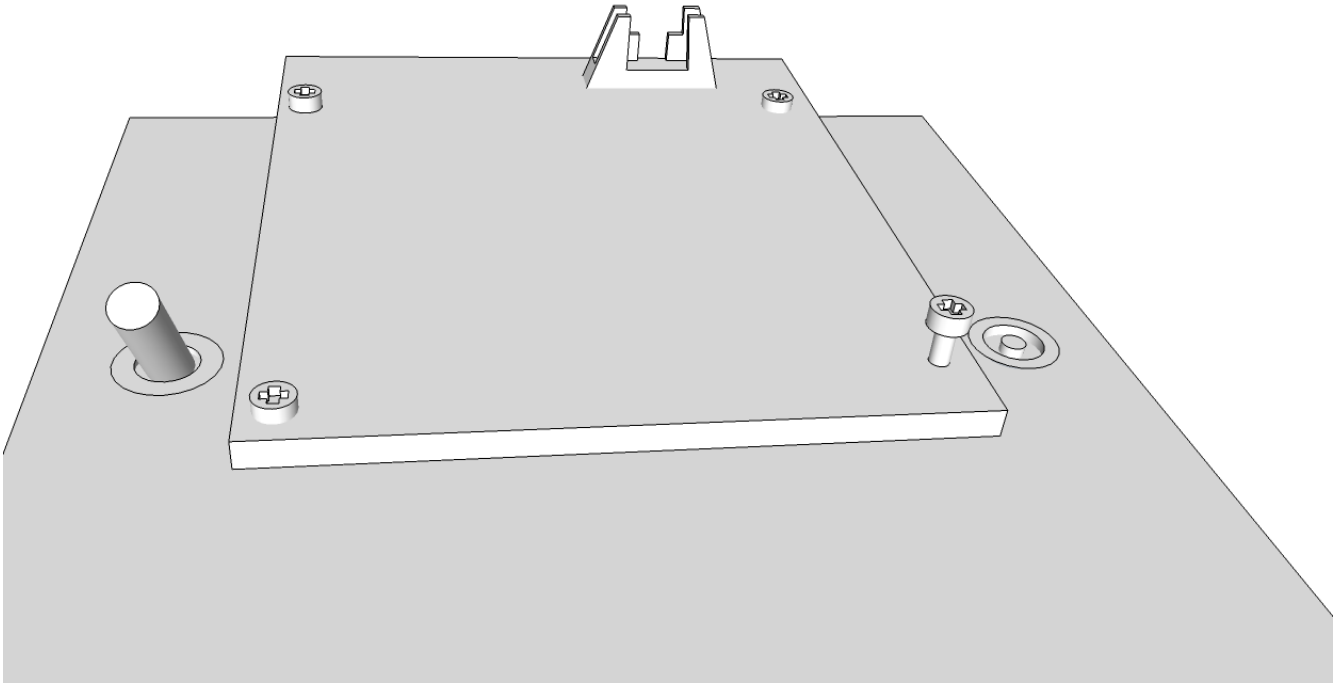


Figure 7.7: Attaching the PCB Brainboard to the Upper Chassis.

Step 5: Attach 2 units of GP2Y0A02YK IR range sensors.

With two metal screws, fasten two metal L-shaped brackets at the front of the top chassis, using the second row of pre-drilled, threaded holes. Twist the brackets along the vertical axis so that they face away from the center line of the chassis, when facing the opponent, forming **V** – shaped lines of sight. Attach IR range sensors to the top of each bracket. Use bolts with matching hexagonal nuts to fasten the range sensors through existing holes in top parts of each bracket. Adjust the metal brackets, as needed, to create an appropriate scanning field for each range sensor.

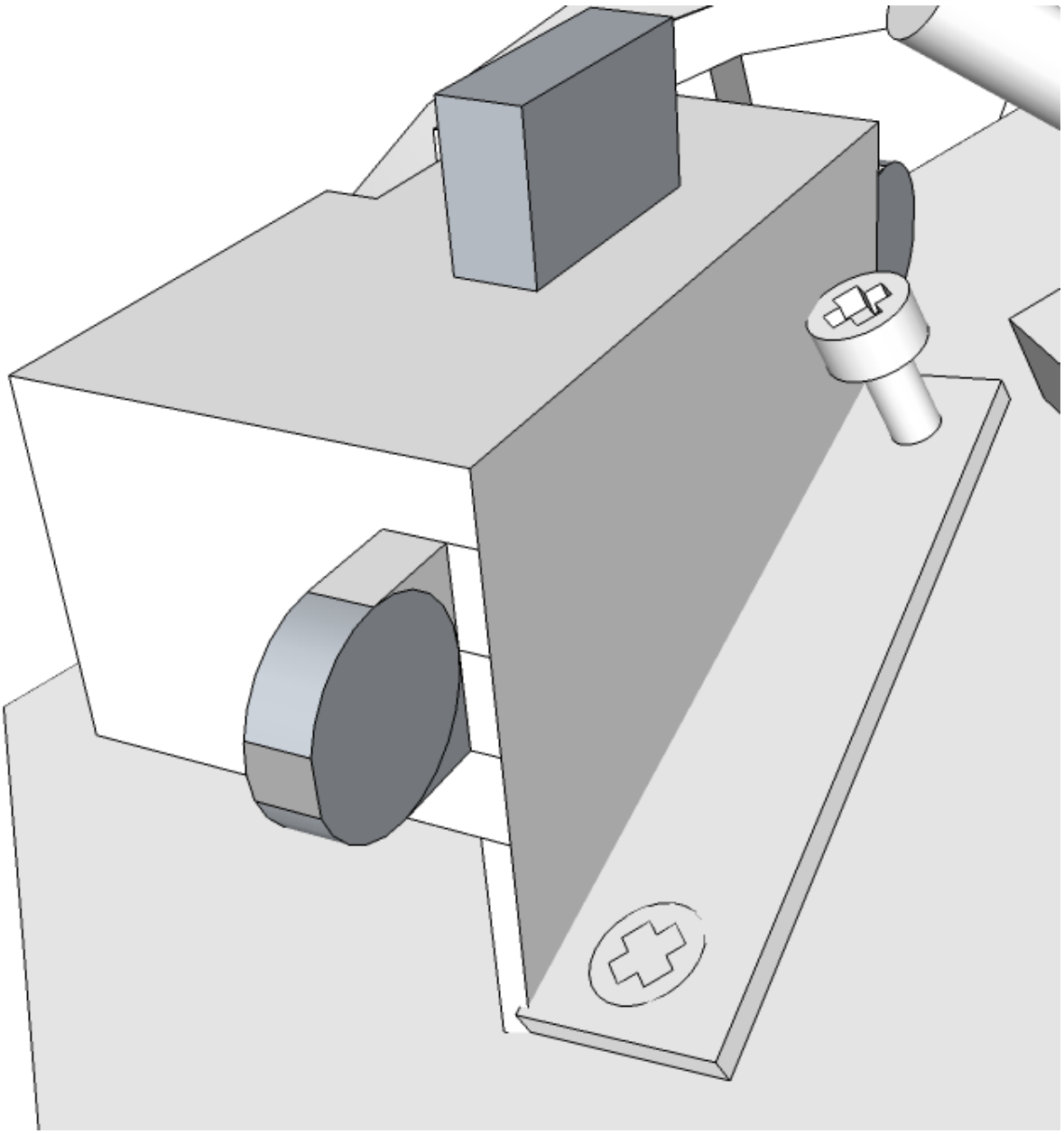


Figure 7.8: Attaching 2 Units of GP2Y0A02YK IR Range Sensors.

Step 5: Attach a Lever-Actuated Contact Switch

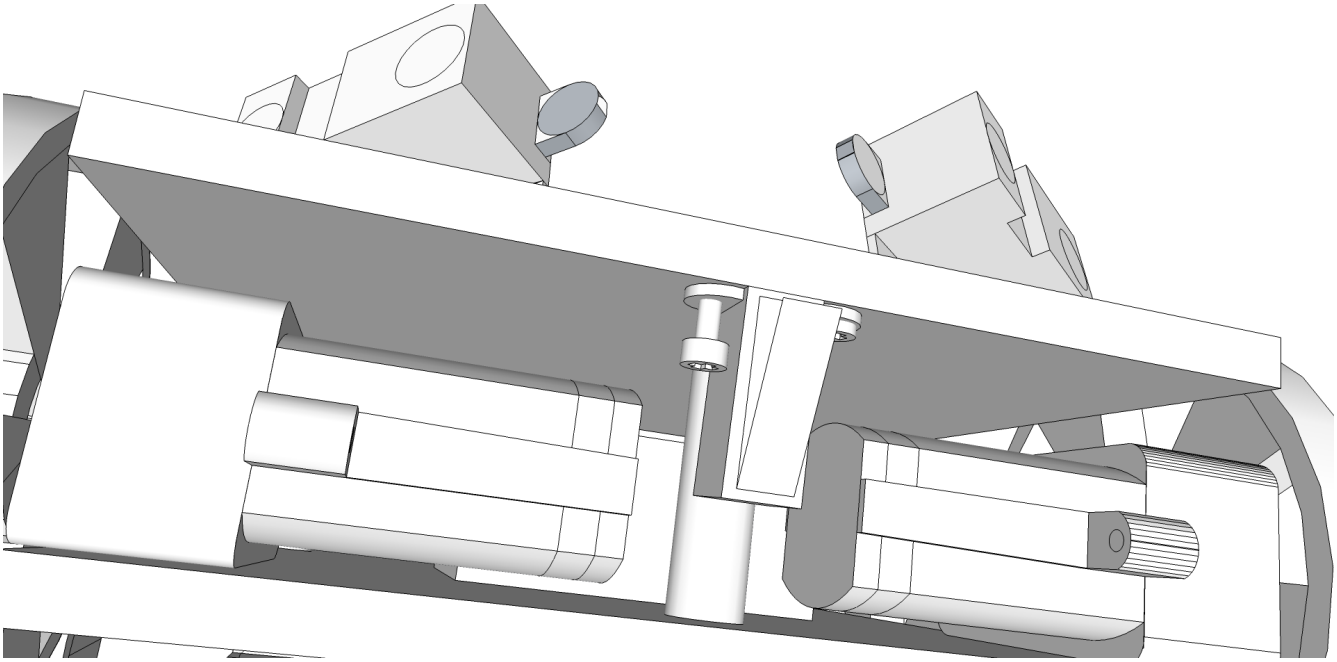


Figure 7.9: Attaching a Contact Switch

Step 6: Attach wheels.

Attach one unit of Solarbotics GMPW wheel to the output shaft of each GM8 motor.

Step 7: Attach Ball Caster

Attach one unit of Tamiya ball caster in the back of the robot (see Figure 5.26 on next page).

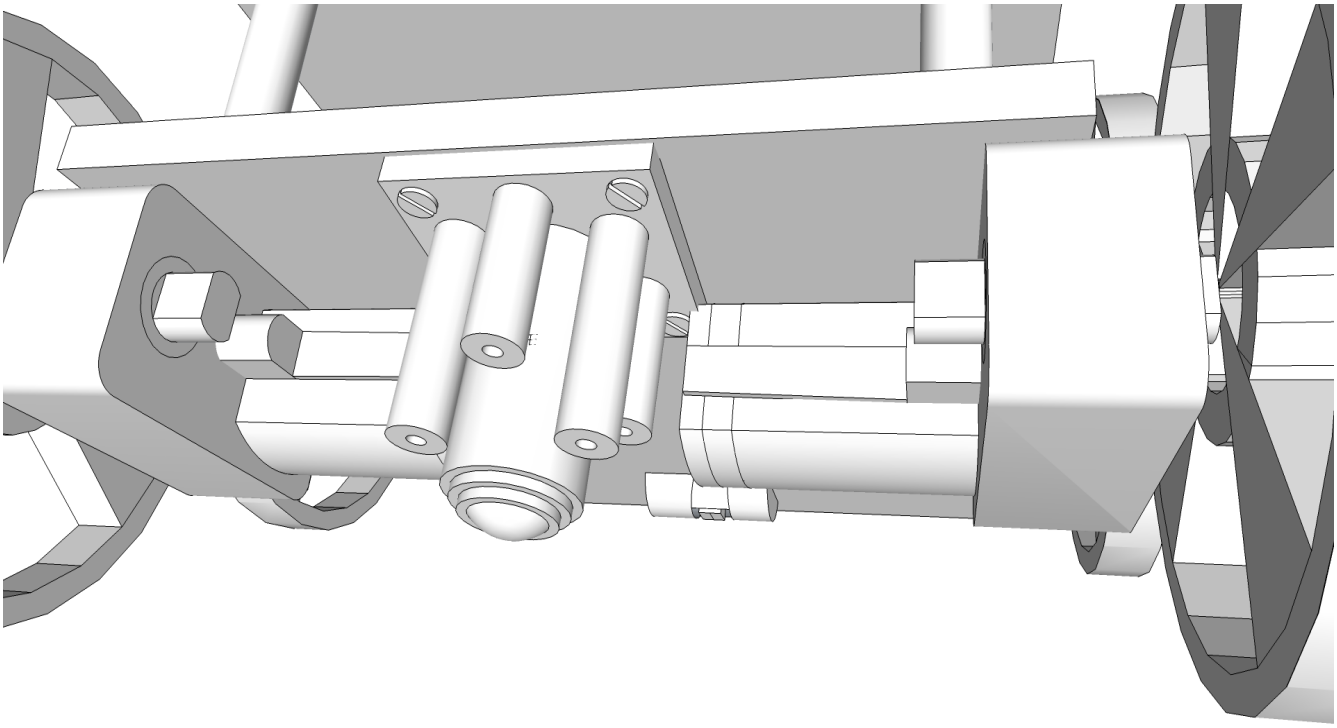


Figure 7.9: Attaching Tamiya Ball Caster to the Rear of the Robot

Step 8: OPB606A IR line sensor.

Use the collapsed U-shaped metal bracket with 1 metal screw, to attach the IR edge sensor to the bottom base platform. Clip the provided square PCB wafers with attached IR edge sensor onto U-shaped metal bracket. Adjust the pliable metal bracket so that the IR edge sensor is suspended approximately 2 mm above the floor.

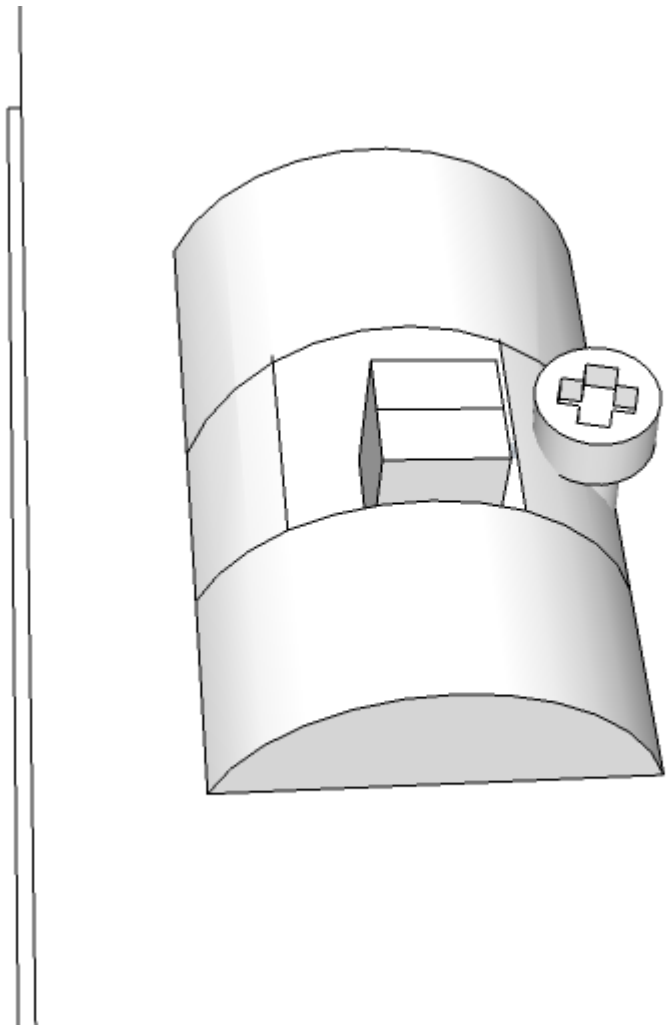


Figure 7.9: Attaching OPB606A IR Line Sensor.

Step 8: Connecting Wires.

Attach connecting wires from motors, sensors, and battery pack to the headers on the PCB brainboard.

8 3-D Technical Drawings of Fully-Assembled SUMO BOT

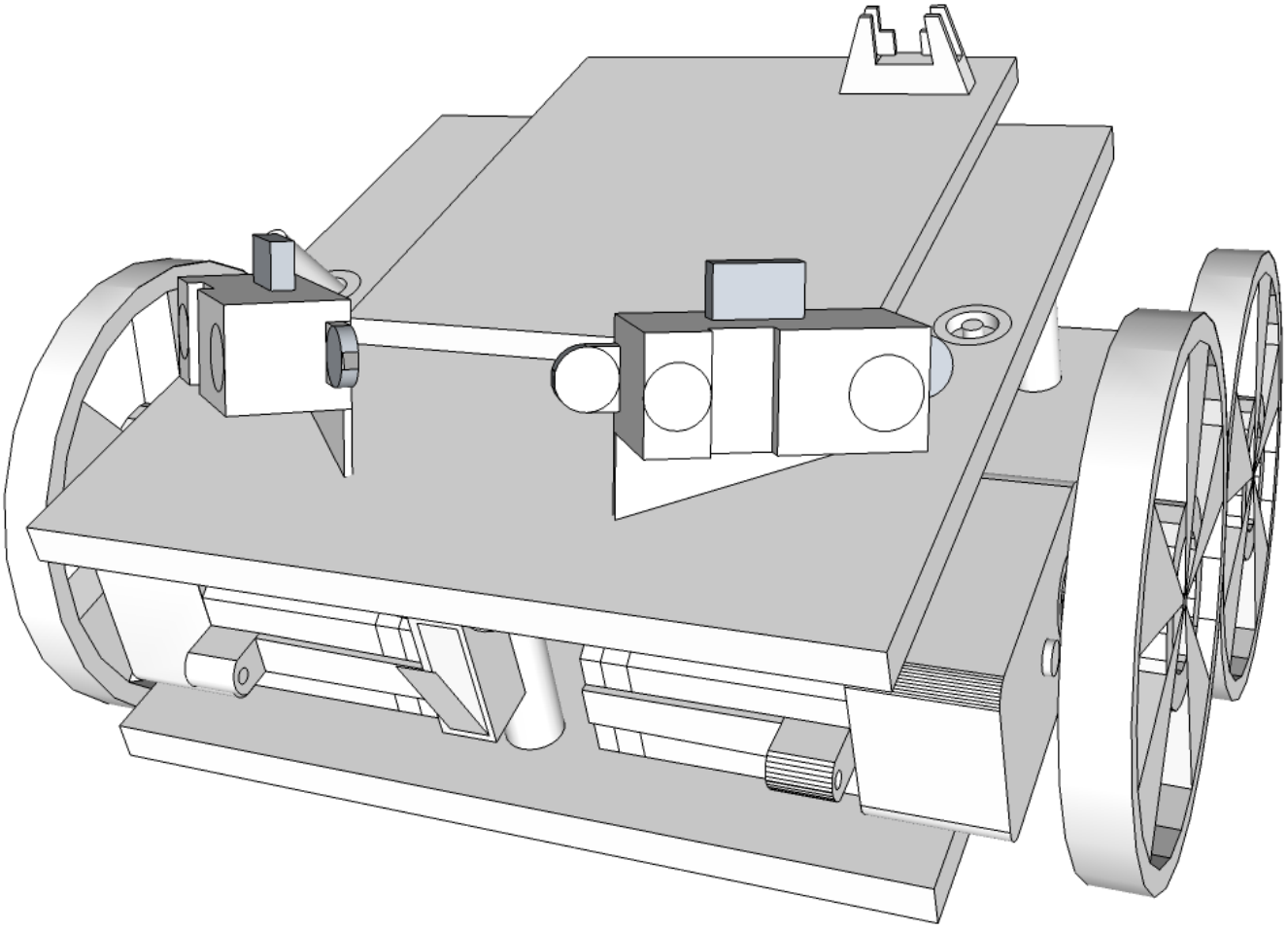


Figure 8.1: Artist's Rendering of a Fully-Assembled Sumo Bot. (Done in Google SketchUp), Front, Diagonal View

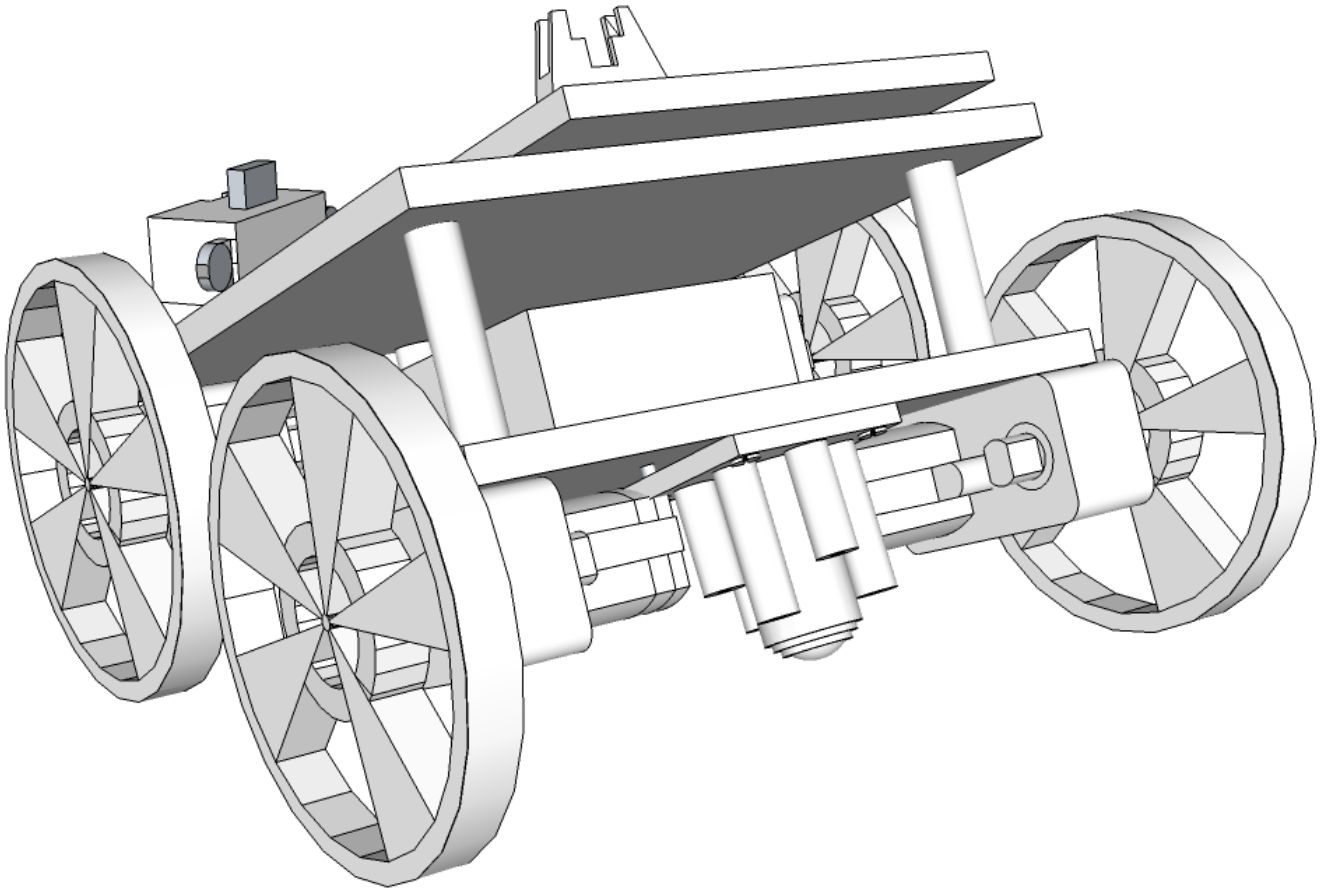


Figure 8.2: Artist's Rendering of a Fully-Assembled Sumo Bot. Back, Diagonal View

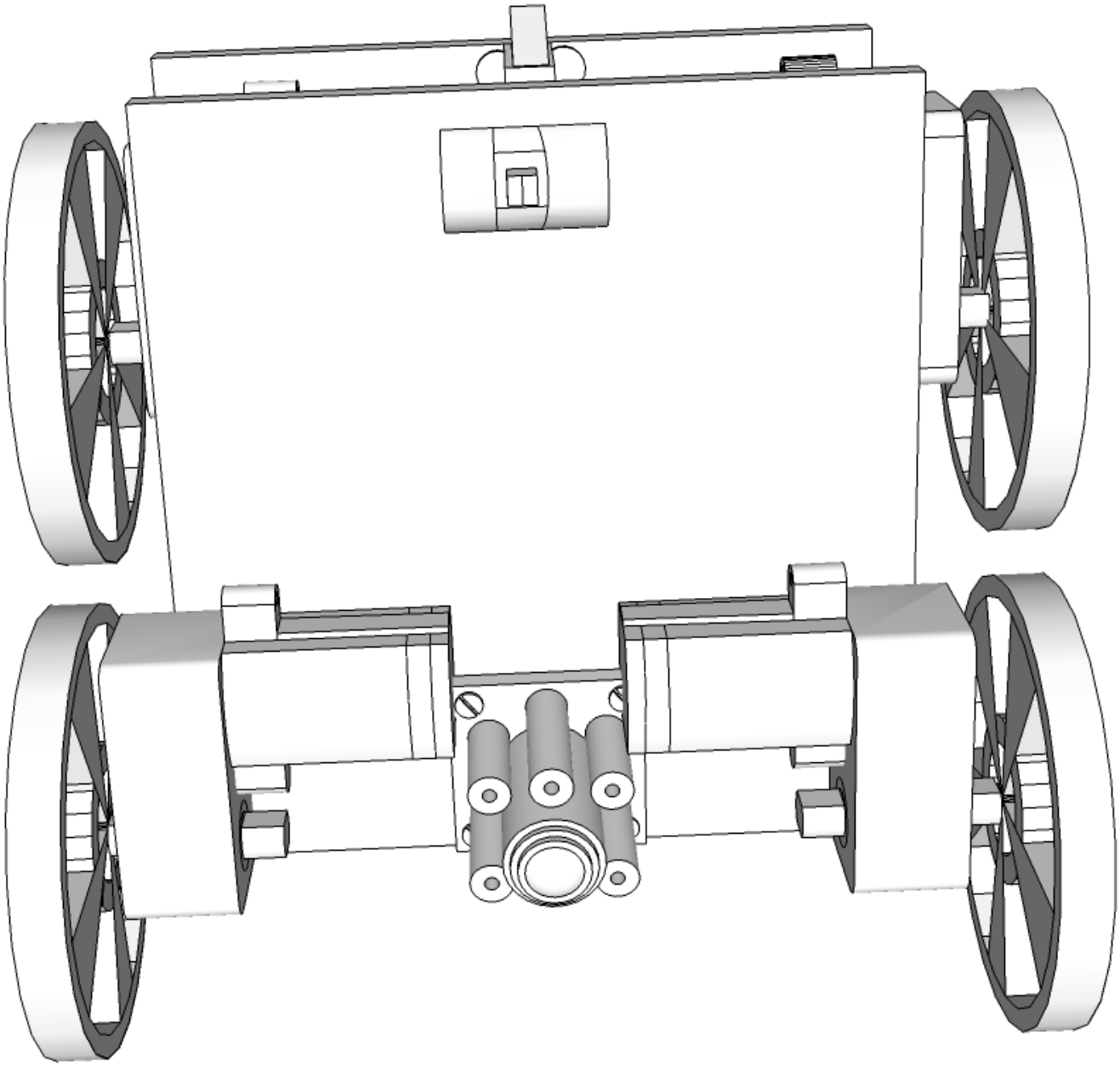


Figure 8.3: Artist's Rendering of a Fully-Assembled Sumo Bot. Front , Underside of the Bottom Chassis View

9 Control Logic, Software

9.1 Pseudocode

```
SIGNAL( ADC reading complete )
{
     if ( reading from left sensor )
     {
          if ( ADC > threshold )
               LeftSensor++;
          if ( front switch is pressed )
               Switch++;
          set to read the right sensor
     }


     else if ( reading from right sensor )
     {
          if ( ADC > threshold )
               RightSensor++;
          set to read the line sensor
     }

     else if ( reading from line sensor )
     {
          if ( ADC > threshold )
               LineSensor++;
          set to read the left sensor
     }
}

void Forward()
{
     RightWheel = full speed;
     LeftWheel = full speed;
}
```

```
void Spin( Side )
{
     if( Side==Left )
          RightWheel = full speed;
          LeftWheel = full speed;


     if ( Side==Right )
          RightWheel = reverse;
          LeftWheel = full speed;
}


void Stop()
{
     RightWheel = stop;
     LeftWheel = stop;
}

void Wait( int time )
{
     int i;
     for ( time )
          Stop();
}

void Backward( int time )
{
     for ( time )
          LeftWheel = RightWheel = Backward;
}
```

```
int main()
{
     while( ! FrontSwitch )
     {}
     Wait( 5 sec );
     Spin( 180 deg, Left );
     Forward();
     while ( forever )
     {
          if ( we have enough (7) readings from sensors and switches
               via the ADC )
          {
               if ( FrontSwitch > 4 )
               {
                    Forward();
               }
               else if ( LeftSensor > ( RightSensor+4 )
               {
                    Spin(Left);
               }
               else if ( RightSensor > ( LeftSensor+4 )
               {
                    Spin(Right);
               }
               else
               {
                    Forward();
               }
               LeftSensor = 0;
               RightSensor = 0;
               FrontSwitch = 0;
          }
          if ( we have enough (4) readings from the line sensor )
          {
               if ( LineSensor >= 3 )
               {
                    Stop();
                    Backward();
                    Spin( 120 deg, Left );
                    Stop();
               }
               LineSensor = 0;
          }
     }
}
```

9.2 Flow Chart of Robot's Behaviour

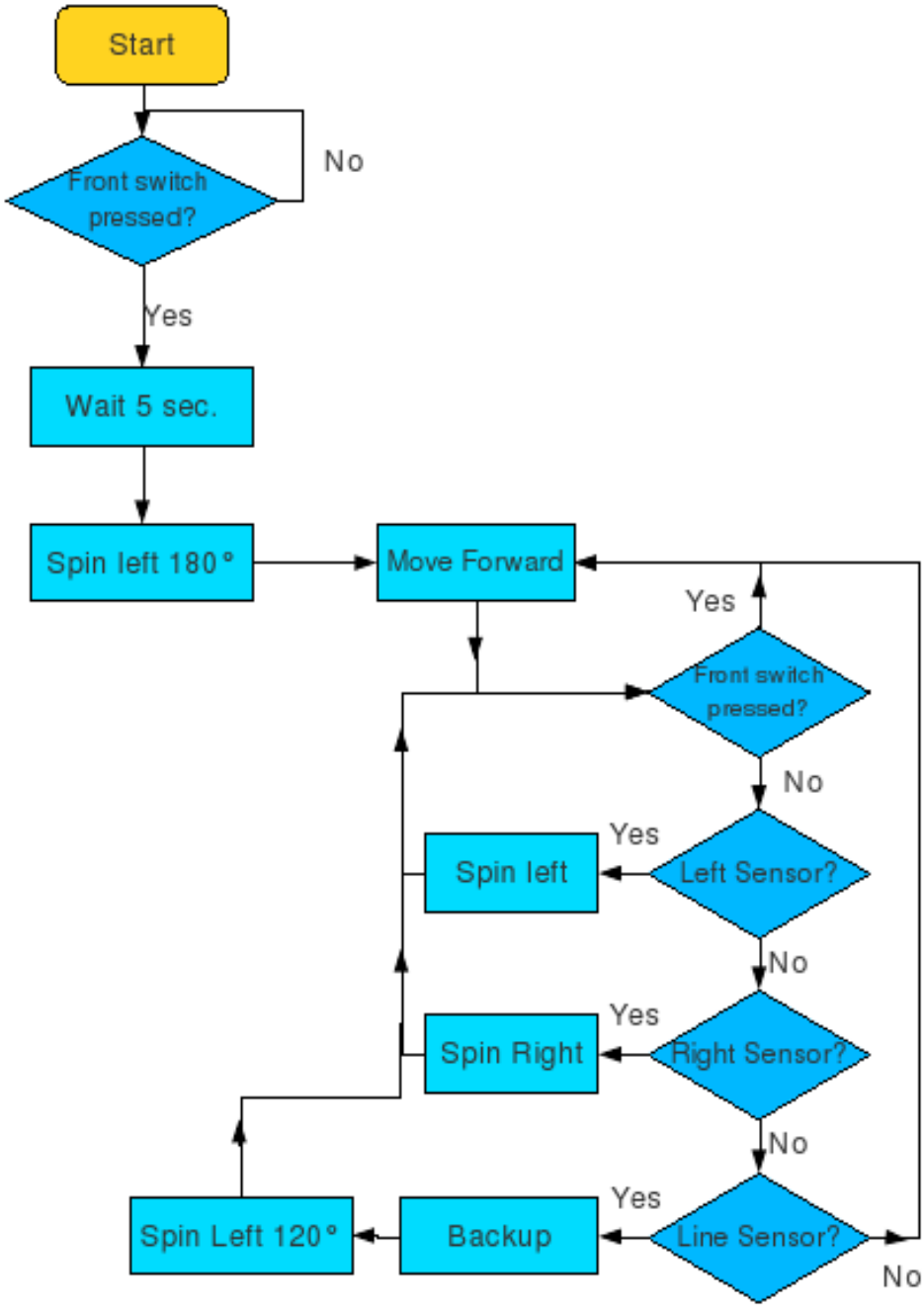


Figure 9.1: Flow Chart of Robot's Behaviour

9.3 Conceptual Algorithm

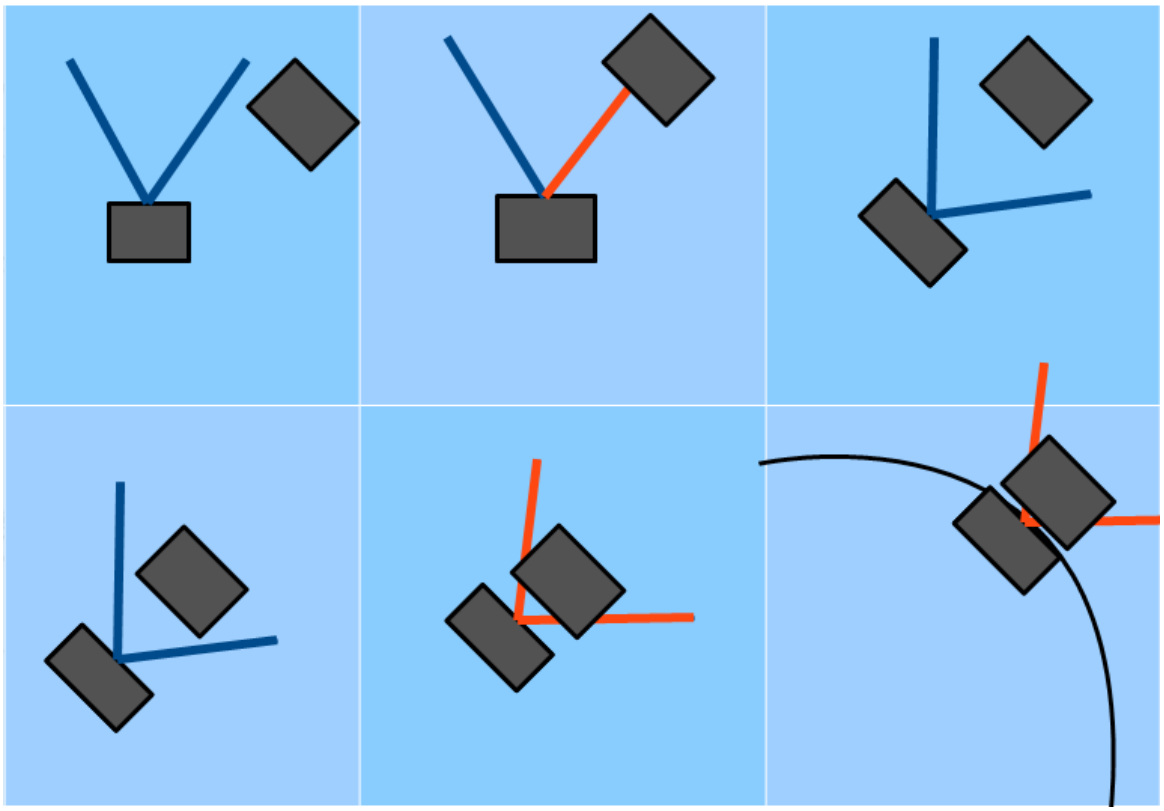


Figure 9.2: Robot Detection Algorithm

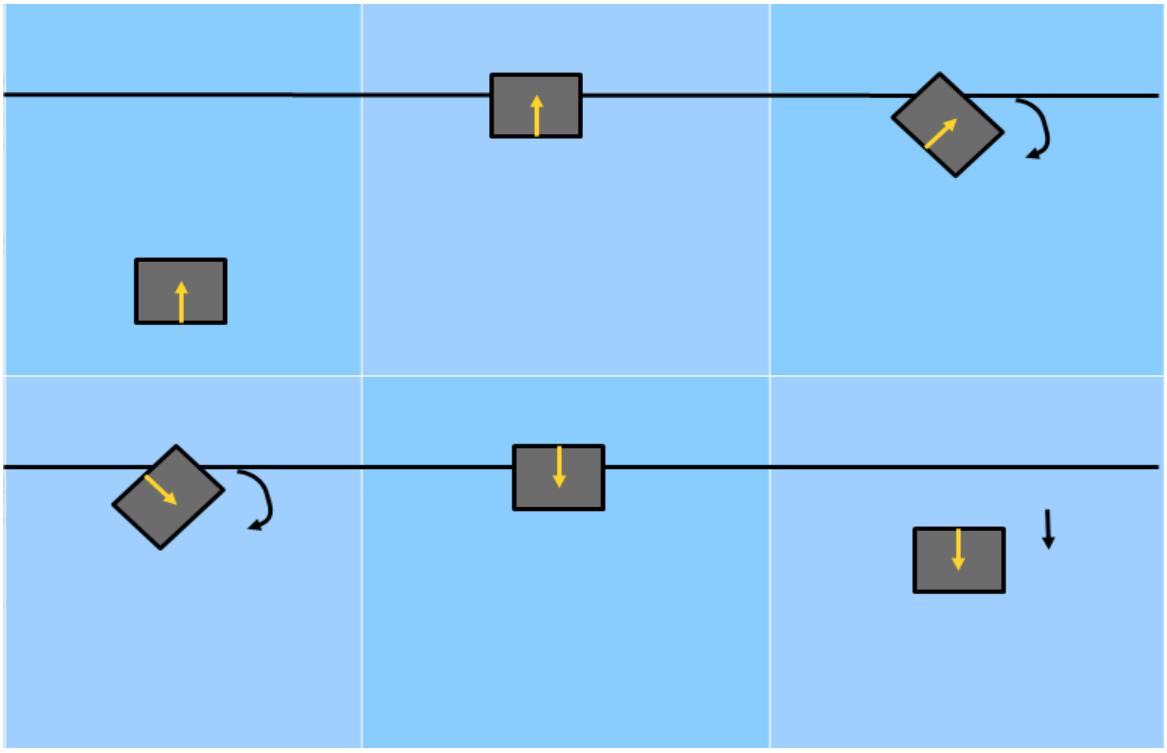


Figure 9.3: Line Detection Algorithm

9.4 Source Code Listing

```
#include <avr/interrupt.h>
#include <stdio.h>
#include <avr/io.h>
#include <stdint.h>

#define Right 0
#define Left 1


int     ASensor=0;
int RightSensor=0;
int     LeftSensor=0;
int FrontSensor=0;
int FrontSwitch=0;

int FrontLine=0;
int BackLine=0;

int Lines=0;

int ADCvalue=0;
int LeftWheel=0;
int RightWheel=0;

int tick=0;
int duty=0;

int ADCState=0;


void SetClock16()
{
        TIFR &= 0b11111011;
        TCNT1 = 0xFFF0;
        TCCR1B |= 0b00000100;
        TIMSK |= 0b00000100;
}

void INIT_ADC()
{
        ADMUX = 0b11000011;
        ADCSRA = 0b11101000;
        ADCState=0;
}

SIGNAL(ADC_vect)
{
        ADCvalue = ADC;
        if (ADCState==0)
        {
                if((PIND & 0b01000000)==0)
                        FrontSwitch+=2;
```

```
                ADMUX = 0b11000100;
                if(ADCvalue>0x0F0)
                        LeftSensor++;
                ADCState=1;
                ASensor++;
        }
        else if (ADCState==1)
        {
                ADMUX = 0b00000001;
                if(ADCvalue>0x0F0)
                        RightSensor++;
                ADCState=2;
                ASensor++;
        }
        else
        {
                ADMUX = 0b11000011;
                ADCvalue=ADC;
                if (ADCvalue>0x2D0)
                        FrontLine++;
                ADCState=0;
                Lines++;
        }
}

SIGNAL(SIG_OVERFLOW1)
{
        if(LeftWheel>0)
        {

                if(duty<LeftWheel)
                {
                        PORTB |= 0b01000001;
                        PORTB &= 0b01111111;
                }
                else
                        PORTB &= 0b00111111;
        }
        else if (LeftWheel<0)
        {
                if(duty<(0-LeftWheel))
                {
                        PORTB |= 0b10000001;
                        PORTB &= 0b10111111;
                }
                else
                        PORTB &= 0b00111111;
        }
        else
                PORTB |= 0b11000001;


        if(RightWheel>0)
        {
                if(duty<RightWheel)
```

```
                        {
                        PORTB |= 0b00000101;
                        PORTB &= 0b11111101;
                        }
                else
                        PORTB &= 0b11111001;
        }
        else if (RightWheel<0)
        {
                if (duty<(0-RightWheel))
                        {
                        PORTB |= 0b00000011;
                        PORTB &= 0b11111011;
                        }
                else
                        PORTB &= 0b11111001;
        }
        else
                PORTB |= 0b00000111;

        tick++;
        duty++;

        if(duty>3)
                duty=0;

        SetClock16();
}

void Turn(int time, int Side) //time in millisec Side Left or Right
{
        tick=0;
        while (tick<((time*10)+1))
        {
                if(Side==Left)
                {
                        LeftWheel=1;
                        RightWheel=4;
                }
                else
                {
                        RightWheel=1;
                        LeftWheel=4;
                }
        }
}

void Forward()
{
        RightWheel=4;
        LeftWheel=4;
}

void Spin(int time, int Side)
{
```

```
        int i;
        for (i=0; i<time ;i++)
        {
                int j;
                for (j=0; j<18; j++)
                {
                        if(Side==Left)
                        {
                                LeftWheel=-4;
                                RightWheel=4;
                        }
                        else
                        {
                                LeftWheel=4;
                                RightWheel=-4;
                        }
                }
        }
}

void Stop()
{
        RightWheel=0;
        LeftWheel=0;
}


void Wait(int time)
{
        int i;
        for (i=0; i<time ;i++)
        {
                int j;
                for (j=0; j<300; j++)
                        Stop();
        }
}


void Reset()
{
        ASensor=0;
        RightSensor=0;
        LeftSensor=0;
        FrontSwitch=0;

        FrontLine=0;
        Lines=0;
}


void Backward(int time)
{
int i;
        for (i=0; i<time ;i++)
```

```
        {
                int j;
                for (j=0; j<100; j++)
                {
                LeftWheel=-4;
                RightWheel=-4;
                }
        }
}


int main()
{
        INIT_ADC();
        SetClock16();
        DDRB = 0b11000111;
        DDRD = 0b10000000;

        PORTD |= 0b01000000;

        while((PIND & 0b01000000)!=0)
        {}

        Wait(370);

        sei();

        Spin(18,Left);
        Reset();
        Stop();
        Forward();



        while(1)
        {
                if (ASensor > 7)
                {
                        if(FrontSwitch>4)
                        {
                                PORTD &= 0b01111111;
                                Forward();
                        }
                        else if ( LeftSensor > (RightSensor+4) )
                        {
                                PORTD |= 0b10000000;
                                LeftWheel=-1;
                                RightWheel=4;
                        }
                        else if( RightSensor > (LeftSensor+4))
                        {
                                PORTD |= 0b10000000;
                                LeftWheel=4;
                                RightWheel=-1;
```

```
                        }
                        else
                        {
                                PORTD |= 0b10000000;

                                LeftWheel=4;
                                RightWheel=4;
                        }

                        cli();
                        ASensor=0;
                        LeftSensor=0;
                        RightSensor=0;
                        FrontSwitch=0;
                        sei();
                }

                if (Lines>4)
                {
                        if (FrontLine>(Lines-2))
                        {
                                PORTD |= 0b10000000;
                                Stop();
                                Backward(2);
                                Spin(12,Left);
                                int i=0;
                                ASensor=LeftSensor=RightSensor=0;
                                int Detected=0;
                                while(!(Detected)&&(i<4))
                                {
                                        if (ASensor>7)
                                        {
                                                if ( LeftSensor > (RightSensor+4)
)
                                                        Detected=1;
                                                else if( RightSensor >
(LeftSensor+4))
                                                        Detected=1;
                                                else
                                                {
                                                        Spin(1,Left);
                                                        i++;
                                                }
                                                cli();
                                                ASensor=0;
                                                LeftSensor=0;
                                                RightSensor=0;
                                                FrontSwitch=0;
                                                sei();
                                        }
                                }

                                Stop();
                        }
```

```
                        cli();
                        FrontLine=0;
                        Lines=0;
                        Reset();
                        sei();
                }

        }
}
```

10 How to Load Software Onto the Microcontroller

Step 1: (Optional) Using AVR Studio, compile the code that you want to load to the chip.

Step 2: Run 'ponyprog.exe', go to Device, in the AVR micro submenu select Atmega8.

Step 3: Go to File, Open Device File, find the binary file you need to load onto the chip. If you compiled your own code, then the binary file will be found in the "default" folder of your AVR studio project folder. The code will appear in hex format in the 'ponyprog.exe' window.

Step 4: Go to Setup, then Interface Setup; make sure the I/O port setup is set to SERIAL, press OK.

Step 5: Plug your interface cable into the USB port of your computer.

Step 6: Plug the other end of the interface cable into the robot's ISP port.

Step 7: Turn ON your SUMO Bot.

Step 8: In the Command menu of 'ponyprog.exe', click on 'Write All (Ctrl-W).

Step 9: The binary code will now be transferred to the Atmega chip that controls your robot.

NOTE: A variety of pre-made routines in binary and C source code format can be downloaded on the SUMO Bot web site. All code is Open-Source and can be modified at will using the AVR Studio software to adapt it to your own robot.

11 Bibliography

[1] “Building a 5 Volt Power Supply, Microcontroller Beginner Kit." Electronic Tutorials, Electronic Kits, Electronic Tutorials, Electronic Hobby Kits, Electronic Kit, Electronic Hobby Projects. http://www.electronicsteacher.com/tutorial/building-a-5-volt-power-supply.php (accessed April 5, 2010).

[2] “Image of SN754410N, quad half-H bridge”." limor. http://www.ladyada.net/wiki/partselector/ic (accessed April 5, 2010).

[3] “GM8 - Gear Motor 8 - 143:1 Offset Shaft « Products « Solarbotics." Solarbotics. http://www.solarbotics.com/products/gm8/ (accessed April 5, 2010).

[4] ¨GMB28 - GM2/GM8 Gear Motor Bracket, Laser-Cut Steel « Products « Solarbotics." Solarbotics. http://www.solarbotics.com/products/gmb28/ (accessed April 5, 2010).

[5] ¨GMPW - GM Series Plastic Wheels « Products « Solarbotics." Solarbotics. http://www.solarbotics.com/products/gmpw/ (accessed April 5, 2010).

[6] ¨Sharp GP2Y0A02YK IR Sensor." Acroname Robotics. http://www.acroname.com/robotics/parts/R144-GP2Y0A02YK.html (accessed April 5, 2010).

[7] "Pololu - Tamiya 70144 Ball Caster Kit (2 Casters)." *Pololu Robotics and Electronics*. Web. 11 Apr. 2010. <http://www.pololu.com/catalog/product/66>.