Dinosaur Lightning Tornado Sumo Robot

ME-353 Mechatronics, 2024

Saira Billah, Azam Khan, Noam Schuck

Introduction

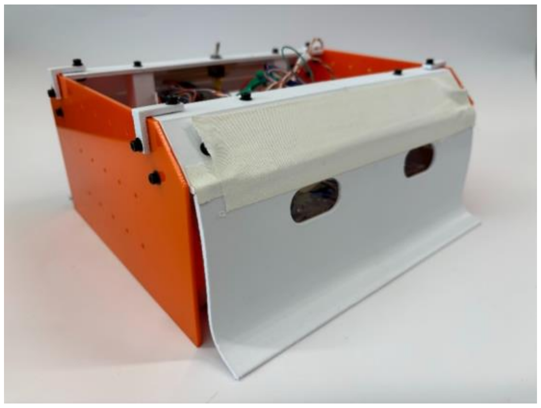


Figure 1.1: The completed Dinosaur Lightning-Tornado

A sumo robot competition is a tournament in which autonomous robots participate in one-on-one battles in a ring, with the objective being to outlast the opponent in the ring. While sumo wrestling robots are not utilized in the real world, many of the technologies implemented in these robots are utilized throughout the engineering industry. For example, autonomous robots and vehicles like trains, buses, and cars are in development worldwide and will revolutionize transportation when they become commonplace. Additionally, to interact with the outside world, the robots will utilize sensors like LiDAR and IR, which are used in many industries. The robot is built in compliance with competition rules and regulations, which can be found in Appendix I.

Commented [SB1]: idt we need this

Fundamentally, the Dinosaur Lightning-Tornado needs to be able to detect the white border that delineates the ring and avoid it. Beyond that, additional functionalities such as being able to detect the enemy are useful in effective offense or defense. Additionally, mechanical aspects of the robot, such as grippy wheels, low center of gravity, strong motors and exteriors ramps can improve chances of success when engaging another robot.

The most notable feature of the Dinosaur Lightning-Tornado is the modular base that allows for a simplified design and mounting process, and flexibility in moving around components. Another notable feature is the custom wheels made from a 3D printed core surrounded by castable silicone, and laser-cut steel hubs. A third feature is the half-linear, half-parabolic ramps on the front and back, that allow for a shallow angle of attack and thus high mechanical advantage, while keeping the footprint of the ramp small.

2. Mechanical Design

The primary design goal was to create a simple and fundamentally sound robot. Often when prototyping, attempting to optimize a design can lead to complications and hinder the functionality of the robot. This robot prioritizes flexibility and adjustability, as well as simplicity of design and manufacturing. This can be seen through the modular design, adjustable IR mounts, and 3D printing optimized designs. All components are fully removable rather than being glued or taped, and all hardware is M3.

The competition strategy was equally simple. Since the robot will not have the ability to sense an opponent, the only controls will be to stay away from the edge of the ring. It does so by using the IR sensor to detect the edge, back up, turn, and then continue forward.

Modular baseplate

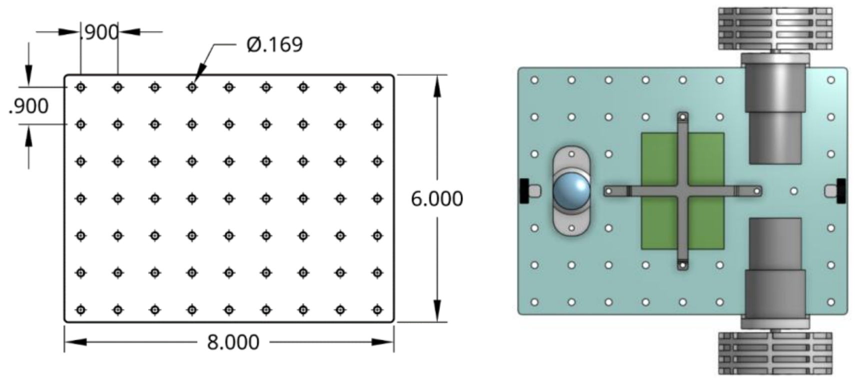


Figure 2.1: Drawing of baseplate (left) with all components on the bottom (right)

The Dinosaur Lightning-Tornado was built on a modular baseplate with evenly spaced holes, which would ultimately simplify design and connection of components, with the additional advantage of being able to freely move parts around after the robot is complete.

The baseplate consists of a rectangular plate with holes spaced 0.9in apart, with each hole being 0.169in, making it large enough for a M3 heat set insert but also small enough to function as a through hole for a M3 bolt with a washer. A 0.9in square can be considered to be one unit, and each component’s sized can be determined by these units; for example, the battery mount (pictured in green and gray in the center of Figure 3b) can be considered to be 5x5, as it spans across 5 holes in each axis. Determining the size of a component in terms of a standard unit simplifies the design process. It also allowed for components to be stacked on top of each other, as well as flexibility in mounting components in different positions, or even to the opposite face of the baseplate.

The baseplate was laser-cut from a 0.1875in piece thick acrylic, and inset with brass threaded inserts using a soldering iron.

Wheels and motors

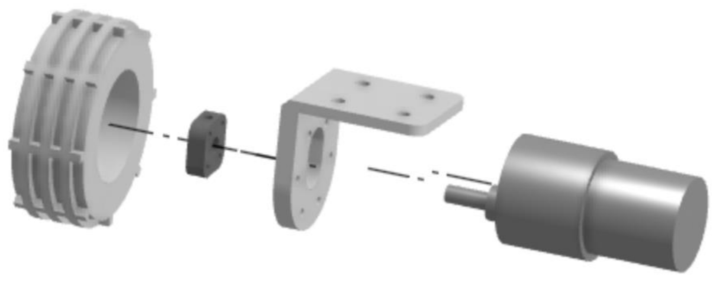


Figure 2.2: Exploded view of wheel core, wheel hub, motor bracket and motor respectively.

There are numerous existing designs and each presents pros and cons. In this design, the robot has three contact points to the ground, consisting of two DC motors connected to two wheels in the back and an additional castor wheel for balance in the front of the robot. While using 4 wheels would allow for additional motors and more overall power (given the current and voltage limits for each motor), having 4 contact points with no-suspension would lead to a statically indeterminate robot when traversing uneven parts of the ring. Using a castor wheel as the third contact point provides a fundamentally simple design, while also simplifying controls for steering and movement.

The wheels are custom made using a 3D printed core with a soft silicone exterior. This creates high-friction wheels that prevent slipping of the robot when a force is applied. Creating custom wheels drastically reduced the cost of making the robot and allowed fine control over the wheel diameter, thickness, design, and even softness of the tread. The wheels were printed alongside a mold for the silicone, and a 2-part silicone was poured around it. The mold needs to be cut off the wheel to avoid damaging the soft silicone. While a two part mold would have allowed for reusable molds, it would have needed to be sturdier and thicker than the single use molds, and ultimately more plastic would have been wasted over the course of the project than with a single use mold for each wheel.

The hubs were laser-cut from 0.25in thick mild steel, with one additional hole for a set screw drilled perpendicular to the laser-cut face using a drill press, and then tapped by hand.

Linear-Parabolic ramp

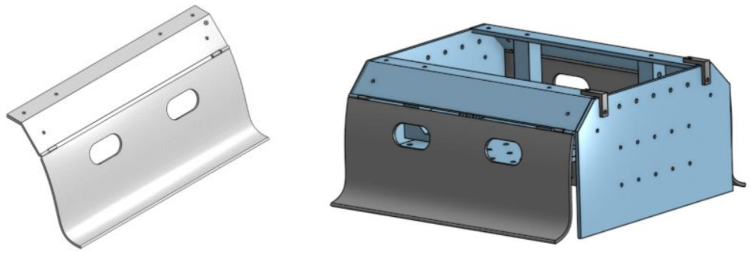


Figure 2.3: Ramp assembly (left), ramps and walls (right)

The front and back of the robot consist of a half linear and parabolic ramp that is connected through a hinge, as seen in Figure 2. This design was chosen as the hinge allows the ramp to sit flush with the ground, and the parabolic section reduces angle of attack. This provides the robot with an mechanical advantage over its competitors by being able to wedge under its opponent. Additionally, this hinge functions as a compliant mechanism to prevent it from replacing one of the wheels as a contact point and lifting the robot when traversing uneven terrain.

Each ramp, linear and parabolic, were 3D printed in two parts, as they were too large for most printers, and were then glued together with superglue and reinforced using wooden sticks glued to the back.

3. Electrical Design

The ATmega328P microcontroller is used to control the robot and manage its peripheries. The black sumo robot ring has a bright white border around the edge. An IR sensor is pointed down and provides data to the microcontroller, which compares the signal it receives from the sensor to the signal it recieves from a potentiometer. The potentiometer is tuned so that its value will be less than that of the IR signal when the IR is directly above the white border. When the IR sensor passes the threshold set by the potentiometer, the microcontroller recognizes that it is at the border of the ring and needs to reverse and turn around. Below the connection and interaction between the electrical components are described in further detail. Additionally, reference the electrical system's block diagram in figure 3.1 and the circuit schematic in figure 3.2 for a better understanding of the structure of the system.

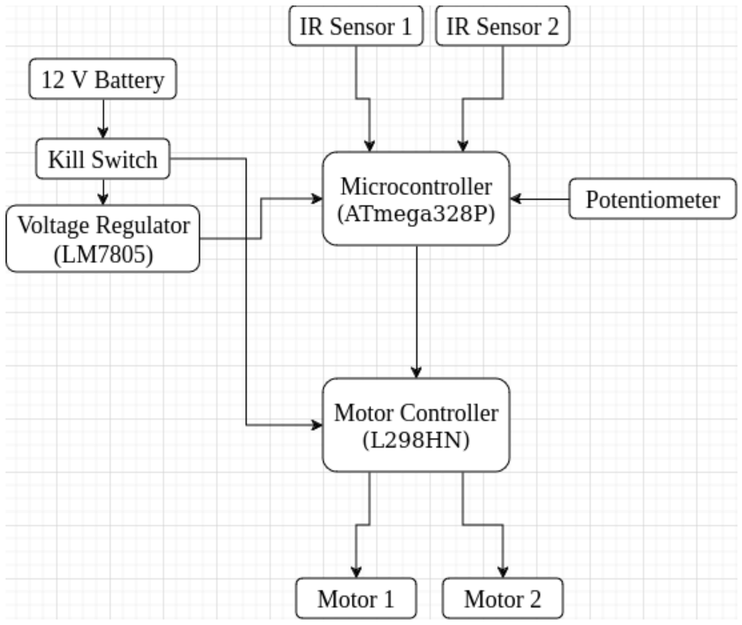

IR Sensor 1
IR Sensor 2
12 V Battery
Kill Switch
Voltage Regulator (LM7805)
Microcontroller (ATmega328P)
Potentiometer
Motor Controller (L298HN)
Motor 1
Motor 2

Figure 3.1: Electrical System Block Diagram

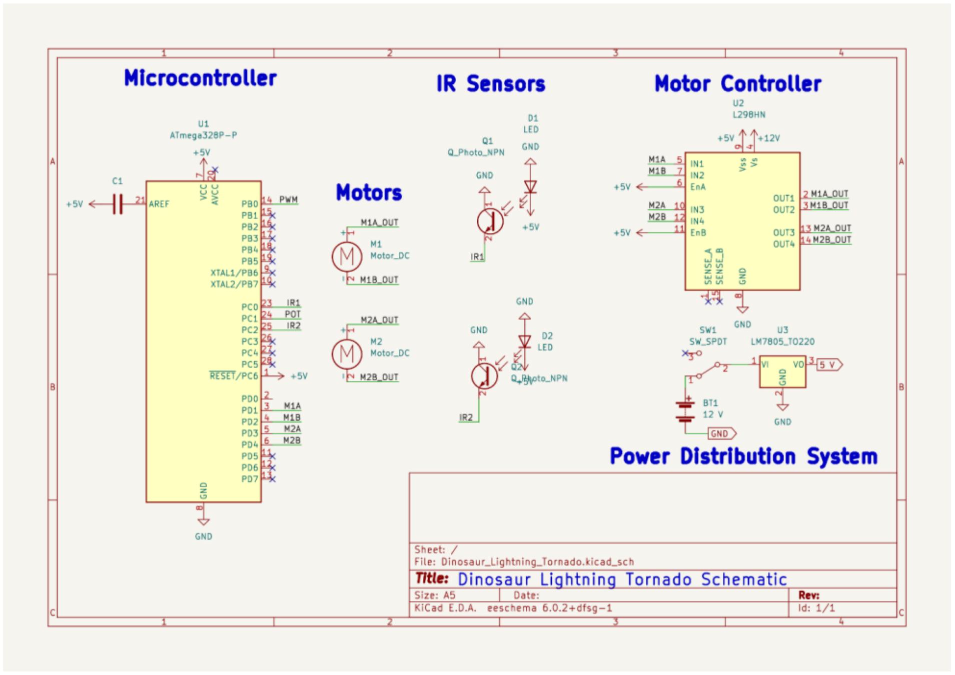


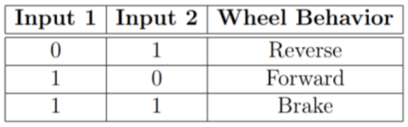
Figure 3.2: Detailed Electronics Schematic

The Battery and Voltage Regulator:

The circuit is powered by a 12 V NiMH battery, which connects, through a single pole double throw (SPDT) kill switch, to the LM7805 voltage regulator and the 74LS298 motor driver. The driver functions on 5 V for its logic pins but also takes 12 V to power the motors. The 5 V that powers the driver, as well as the rest of the circuit components, comes from the voltage regulator, which steps down from 12 V to 5V.

The IR Sensor and Potentiometer:

As mentioned previously, the IR sensor, positioned within a few centimeters from the ground, emits light from an infrared LED, which gets reflected and received by a phototransistor. That analog signal than gets transmitted to the microcontroller from processing. The potentiometer is used as a reference voltage for comparison by the microcontroller with the signal from the IR sensor. The potentiometer is tuned such that when the IR sensors receive light reflected off the white border, the potentiometer voltage will be less than the IR sensor voltage and more than the light the IR sensor receives upon reflection on the black ring. There are two IR sensors, which are connected to pins 0 and 2 on Port C of the microcontroller, and one potentiometer, connected to Port C pin 1. Those pins are among the few pins on which the microcontroller can perform analog to digital conversion (ADC) on.



Input 1	Input 2	Wheel Behavior
0	1	Reverse
1	0	Forward
1	1	Brake

Figure 3.3: Motor Controller Truth Table

The Motor-Controller and Motors:

The motor-controller chip, the 74LS298, is provided with two sources of power: 5 V to power the logic pins and 12 V to power the motors. For each motor, there is one enable pin, two input pins, and two output pins. The enable pin is constantly connected to 5 V power, so the only way to stop the motors from spinning is to activate the breaks or hit the kill switch. The logic for the inputs is displayed above in figure 3.3, and the motors behave as follows. A 0b10, 0b01, and 0b11 spin the motors forward, backward, and activate the breaks, respectively. The outputs from the driver go to each motor, which then spin, propelling the car. Rather than rotating the wheels left and right, the car turns by braking one wheel and reversing the other, creating a rotation centered about the stationary wheel. To vary the speed of the car, a PWM signal generated from Port B pin 1 of the microcontroller is generated and can be switched with the 5 V connected to the enable pin on the motor controller, but as speed control is not necessary for the purposes of the competition, the PWM signal is left disconnected.

4. Program Logic

The ATmega328P was programed using Microchip’s snap programmer, which connects to a computer and then directly to the microcontroller. The program was written in C, utilizing AVR libraries and the XC header library. The flow of the code went as follows (written in psudocode):

```
declare outputs
setupPWM()
setupADC()
setupInterrupt()

go straight

while()
    threshold = adc(potentiometer)
    ir1 = adc(ir1)
    ir2 = adc(ir2)

    if delayed enough time and ir1 and ir2 < threshold
        if reversing
            stop right wheel
            reverse left wheel
        else
            go straight
        reset delay counter
    else if ir1 or ir2 > threshold
        reverse
```

Because an encoder was not used to track the movement of the robot, delays were necessary to turn and reverse. Instead of using the ms_delay() function, which is a type of blocking delay and would stop all microcontroller operations until the time specified passed, the delay was implemented via timer 0 interrupt with a prescaler of 1024. During the interrupt service routine (ISR), the global delay variable is incremented if the robot was supposed to be delaying. If the Dinosaur Lightning Tornado wasn’t turning or reversing, the ISR would still be called but the delay variable would not be incremented. To view the full, well commented program, refer to Appendix IV[1].

[1] The program can also be found at: https://github.com/fakebakon76/ME353/tree/master

5. Discussion

Assembly

Assembly of robot was simple and there were no major complications. When assembling the robot, some final changes were made, including removing one leg of the battery mount and mounting it to the underside of the baseplate, shifting the center of mass of the robot lower and further forward, making it harder to push and less likely to lift the front wheels under acceleration. Additionally, the motors were shifted back to lengthen the wheelbase of the robot for improved stability.

Performance

The Dinosaur Lightning-Tornado fulfilled the requirements of staying within the ring and performed well in informal testing against competitors. The combination of wheels and motors made the robot relatively powerful. The motors were chosen to be as close as possible to the 3A current limit of the drivers, and each motor had a stall current of 2.6A with an operating speed of 150RPM. The drivers were supplied with power from a 12V Ni-Mh battery pack. The wheels are 3in in diameter and had a high coefficient of friction with the mat. The combination of RPM and wheel diameter provided a reasonable speed and enough torque to push other robots around.

Fixes

Despite performing well, the robot had two major issues. The first was that specific circumstances could lead to it leaving the ring on its own. When the Dinosaur Lightning-Tornado was in a corner, it had the potential to detect a border and begin to reverse, but that reversing motion could carry it over a different border. Some solutions for this would be to be to reduce the backing up motion, or remove it entirely and increase the turning motion.

The second major issue was that, despite the hinge mechanism, the ramp at the rear could still sometimes catch on the neoprene mat and lift the wheels off the ground. The solution to this would be to add smoother contact points with the ground to the ramp, for example, by insetting ball bearings into the plastic on the bottom side of the ramp. Another solution would be to raise the ramps slightly, however this would reduce their efficacy in getting under opponents.

The assembly process included some minor issues that could be ameliorated if future robots were to use a similar design. It is important to account for the different tolerances when working with additive and subtractive manufacturing. Holes in particular require tighter tolerances, and need to be oversized when 3D printing, and undersized when laser cutting. This was seen in some of the holes in the hubs, which were incorrectly sized and difficult to tap without breaking the tool.

Another issue was choosing 3D print hinges. A sample of the ramp was 3D printed to test the tolerance of the design. In this, the design moved very smoothly with little friction. The parabolic ramp has a cylindrical extrusion with a whole in place, while the linear ramp has a cylindrical extrusion with a tiny pole sticking out. In theory, these pieces should fit into each other and cause a smooth rotation. However, during the construction process of the robot, the hinge design broke off the ramps due to the supports. Because of this, tape was used to connect the linear and parabolic ramps to each other. This allowed for the free movement of the parabolic ramp, and the same desired effect was achieved.

Another issue with the ramps is that the size is too large for most 3D printers to be able to print. To mediate this issue, the ramps were printed in halves and superglued together. Flat wooden sticks were used to support the back of the ramps and prevent the halves from bending.

6. Conclusion

In conclusion, the Dinosaur Lightning-Tornado fulfils all requirements and is a strong competitor. The design philosophy was well incorporated into the final product, with the parts being easy to create and the robot simple to put together, with a lot of flexibility to make changes at any step of the process. This was also reflected in the program logic, which was simple and streamlined but effective.

Appendix I: Sumo Robot Rules and Regulations

Spirit of the Problem:

The goal of the competition is for one robot to push the other robot out of the “ring”.

Competition Rules

1. The “ring” will be approximately 3ft by 3ft square, and made from neoprene rubber (mcmaster # 9455K62 - ASTM D2000 BC).
2. The opponents will be placed at one of three starting positions (straight on, 45 degrees, and 90 degrees from opponent) within the ring.
3. When signaled, the teams will activate their robots simultaneously.
4. The robot will attempt to locate the other robot and push it out of the ring.
5. The robot is considered “out of the ring” when more than half of the robot is no longer within the main area of the ring.
6. If neither robot has won after 1 minute (or when both teams agree on a draw), the round is considered a draw.
7. Bumps and imperfections will be less than 1/8” at any single spot and no more than 1/2” over the entire area.

Cost, Size, and Weight

1. The total cost of the robots components may be no more than $200.
2. The cost is calculated as 1/100 the cost to build 100 robots. (NOTE: the cost is NOT what you pay. It is a quoted cost of materials.)
3. Materials donated, found, etc must be accounted for at quoted prices for the total cost of the robot.
4. Basic electronics package can be assumed to be valued at $25
5. The robot may be no larger than 10"x10"x6"
6. The robot may weigh no more than 5 lbs (with all batteries, motors, etc.)

Motors

1. Motors may not have a stall current of more than 3A (each).
   - NOTE: Stall current must be documented.

Power supply

1. The motor power supply must be a combination of AAA, AA or C cells, providing a maximum of 14V (they must be NiCd or NiMH).
2. The power supply for electronics may be at the team's discretion, and may feed from the same power supply as the motors.

Electronics

1. The robot must be controlled by one or more ATMega328P. No other programmable chip is allowed.
2. The robot will sense the border of the ring with one or more supplied IR sensors.
3. The robot may (though not necessarily) employ additional sensors, a LIDAR sensor will be included in the kits – any additional sensors must be accounted for in the total cost.
4. No external control of any sort is allowed.
5. The robot MUST have an easy to access kill-switch. This MUST cut ALL current to ALL parts (motors, electronics, logic etc.)

Restrictions

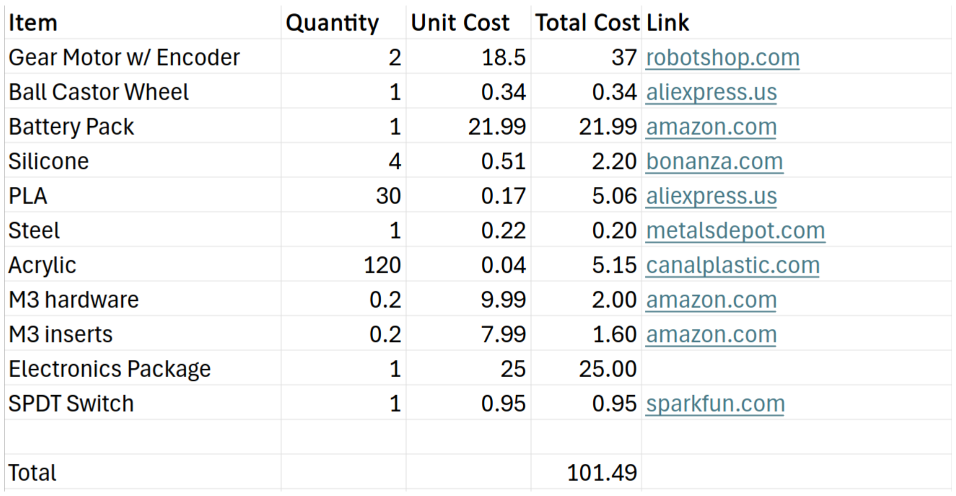
1. No liquids, gels, compressed gases, or hazardous substances may be employed (except within circuitry, NiCd, or NiMH batteries. No lithium-ion or lead acid batteries)
2. No electrical or mechanical weapons are permitted.
3. No lasers may be employed. (This includes sensors.)
4. The robot may not purposefully damage the other robot (this includes reasonably foreseeable "accidents").
5. You may not damage the ring in any way. (this includes, but is not limited to, residue from glue or tape)

On Site Documentation

1. Manifest of all parts and materials.
2. MSDS sheets for all materials.
3. RoHS status for all circuity.
4. Documented amperage limits of drivers.

5. Documented maximum amperage draw for motors.

Appendix II: Parts Listings and Bill of Materials



Item	Quantity	Unit Cost	Total Cost	Link
Gear Motor w/ Encoder	2	18.5	37	robotshop.com
Ball Castor Wheel	1	0.34	0.34	aliexpress.us
Battery Pack	1	21.99	21.99	amazon.com
Silicone	4	0.51	2.20	bonanza.com
PLA	30	0.17	5.06	aliexpress.us
Steel	1	0.22	0.20	metalsdepot.com
Acrylic	120	0.04	5.15	canalplastic.com
M3 hardware	0.2	9.99	2.00	amazon.com
M3 inserts	0.2	7.99	1.60	amazon.com
Electronics Package	1	25	25.00	
SPDT Switch	1	0.95	0.95	sparkfun.com
Total			101.49	

Appendix III: CAD Drawings

Appendix IV: Code Used to Program the ATmega328P

```
/* Pins:
*      Port D      - Output
*      Port D, 1-2 - Motor 1 Control
*      Port D, 3-4 - Motor 2 Control
*      Port C, 0   - IR Sensor Front
*      Port C, 1   - Potentiometer
*      Port C, 0   - IR Sensor Back
*      Port B, 1   - PWM For Motors
*/

#include <xc.h>
#include <avr/io.h>
#include <avr/interrupt.h>

#define SPEED_PIN 1
#define M1A 1
#define M1B 2
#define M2A 3
#define M2B 4
#define POT 1
#define IR1 0
#define IR2 2

void setup_ADC(void);
void setup_PWM(void);
void setup_timer(void);
unsigned int get_ADC(unsigned char);
void change_bit(volatile uint8_t *, unsigned char, unsigned char);
void change_direction(unsigned int duration, unsigned char direction, unsigned char synchronized);

unsigned int threshold = 0, ir1 = 0, ir2 = 0, delay = 0, target_delay = 0, in_reverse = 0;

int main(void) {
DDRD |= (1 << M1A) | (1 << M1B) | (1 << M2A) | (1 << M2B);
DDRB |= (1 << SPEED_PIN);

setup_ADC();
setup_PWM();
setup_timer();

OCR1A = 128; // Sets PWM to 50% duty cycle
change_direction(0, 0b10, 2);   // continue going straight

// Occurs from now till infinity
while(1) {
threshold = get_ADC(POT);                        // gets ir1 of pot threshold
ir1 = get_ADC(IR1);                              // Get the result of the IR ADC
ir2 = get_ADC(IR2);                              // Get the result of the IR ADC

if(delay > target_delay && ir1 < threshold && ir2 < threshold)
        {
if(in_reverse) {
                change_direction(0, 0b11, 1);
                change_direction(250, 0b10, 0);
                in_reverse = 0;
            } else
                change_direction(0, 0b10, 2);   // continue going straight

            target_delay = 0;
delay = 0;
}
        else if(ir1 >= threshold | ir2 >= threshold) // Runs only if we aren't delaying
        {
            change_direction(1000, 0b01, 2);
            in_reverse = 1;
        }
}
}

// Sets up ADC
void setup_ADC(void)
{
ADCSRA = (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0); // Enable ADC | Set Prescaler Bits (0-2) to 111 (128 ticks of system clock is 1
tick for the ADC (i think))
ADMUX = (1<<REFS0);                                    // Set voltage reference to AVcc with external cap at AREF pin
}

// Sets up timer and interrupt
void setup_timer(void) {
TCCR0A |= (1 << COM0A1); // Set timer mode to clear on compare
TCCR0B |= (1 << CS02) | (1 << CS00); // Set timer prescalar to 1024
OCR0A = 10;    // Set compare value to 97 ticks (976 ticks in 1 second with the prescalar set to 1024) as clock is at ~ 1MHz
TIMSK0 = (1 << OCIE0A);  // Enable interupt
sei();                   // Enable global interrupt
}

ISR(TIMER0_COMPA_vect) {
if(target_delay) delay++;
}

// Sets up PWM for Port B1
void setup_PWM(void) {
TCCR1B  |= (1 << CS12); // Sets Prescaler for timer 1 to 256
TCCR1A  = (1 << COM1A1) & ~(1 << COM1A0); // Defines that when a match occurrs to update the PWM when the timer goes to 0 again
TCCR1A |= (1 << WGM12) | (1 << WGM10); // Turns on fast PWM mode
}

// Gets the ADC value from analog pin channel
unsigned int get_ADC(unsigned char channel)
{
ADMUX = (ADMUX & 0xF8)|channel;  // Do ADC on channel pin
change_bit(&ADCSRA, ADSC, 1);    // Start ADC conversion
while(ADCSRA & (1<<ADSC));       // While ADC is still in progress, do nothing
return ADC;                      // Return the ADC value
}

// Is used to start the ADC conversion -> make parameters a reference to (&) ADCSRA, ADSC, and 1, respectively
void change_bit(volatile uint8_t *SFR, unsigned char my_bit, unsigned char bit_val)
{
if(bit_val == 0) {
*SFR &= ~(1 << my_bit); // If my_bit is ADSC then it will stop ADC conversion
} else {
*SFR |= (1 << my_bit); // If my_bit is ADSC then it will start ADC conversion
}
}

/* Changes the direction of the robot
*  Parameters:
*     duration     - how long to turn for in ms
*     direction    - determines motor operation given by motorcontroller truth table
*     synchronized - if 0 targets left motor, if 1 targets right motor, if 2 targets both
*/
void change_direction(unsigned int duration, unsigned char direction, unsigned char synchronized) {
if(synchronized == 2) {
PORTD &= ~((~direction << M1A) | ~(direction << M2A)); // AND it to get the 0s in place
PORTD |= (direction << M1A) | (~direction << M2A);     // OR it to get the 1s in place
} else {
PORTD &= synchronized ? ~(~direction << M1A) : ~(~direction << M2A); // Writes the 1s in the direction
PORTD |= synchronized ? (direction << M1A) : (direction << M2A); // Writes the 0s in the direction
}


target_delay = duration/333; // I divide by 333 so that the 1 duration is 1 ms
TCNT0 = 0;
    delay = 0;
}
```