**LAB REPORT: Final Project**

Brandon Ackerman – RBT173

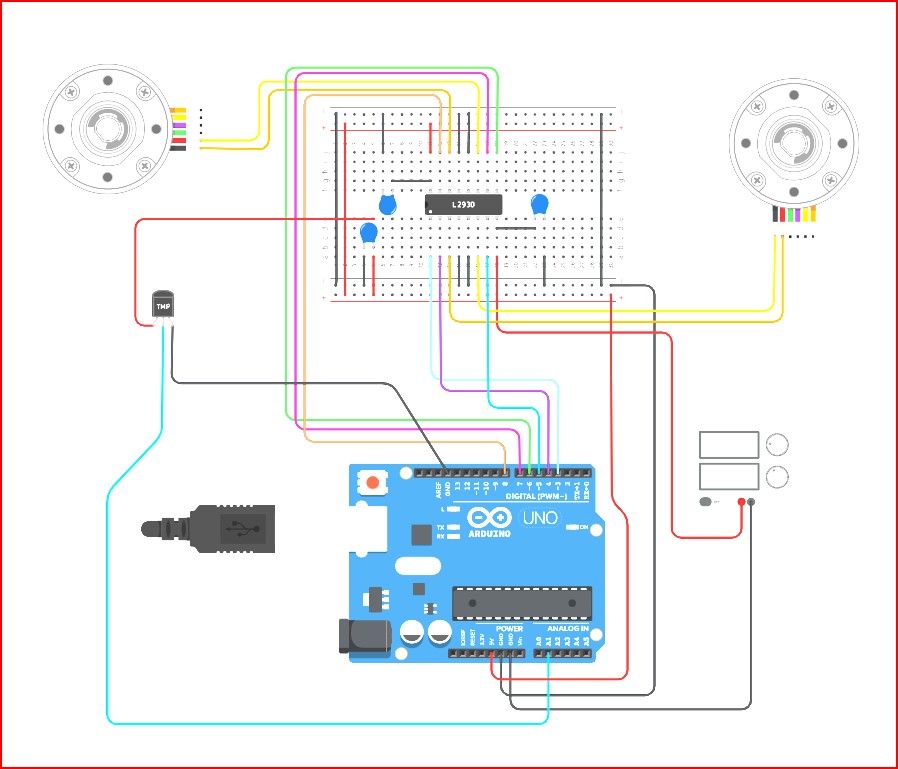
**Introduction**

For our final project, we are tasked with a three-part assignment encompassing everything we have learned throughout our time in RBT173. Given the option of ultrasonic distance sensing and temperature sensing, I opted for the latter. Phase 1: Create a robot capable of stopping before a collision with an obstacle or stopping at a certain temperature threshold. Phase 2: Equip the robot with speed control based on the proximity (or temperature increase) of an object. Phase 3: Provide the robot with limited autonomy by implementing a program that will make the robot reverse, turn, and continue forward, avoiding the object in its path.

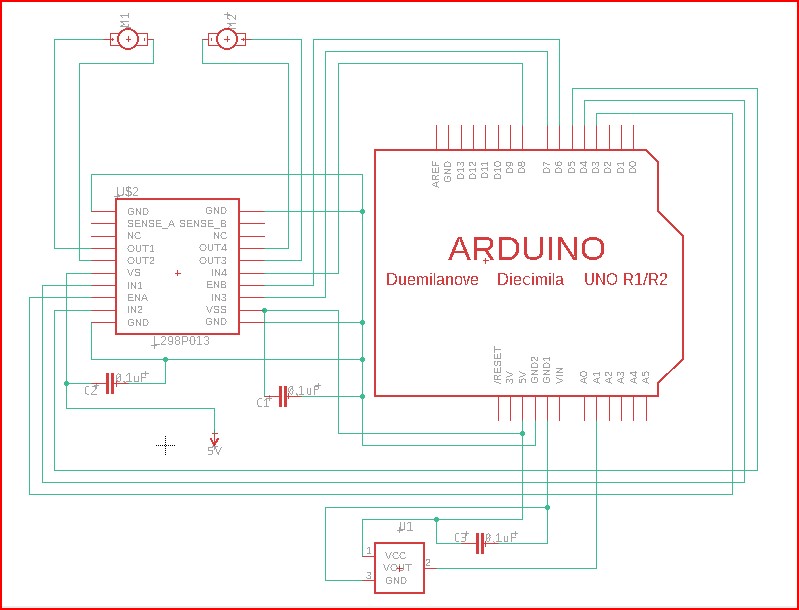
**Results**

The most difficult aspect of the project has been figuring out how to control the speed of the robot as it approaches the obstacle. I am already at a disadvantage, as my family and I have been out of town visiting, so time and resources have been less than ideal. Using TinkerCad and the motor control circuit from our other project, I integrated the TMP36 temperature sensor as an analog input on A1, creating a function that could be called on to read the temperature on a recurring basis. One change of note that I made to the circuit, is the implementation of bypass capacitors. I am not sure if RFI (noise) is even a factor in TinkerCad, but just in case, I added 0.1µF ceramic bypass capacitors on the voltage inputs for the L293 and the TMP35. A quick google search gave me the formula that I needed to convert the mv readings from the sensor into legible temperature measurements in Celsius. To improve my ability to debug the code, I included statements to print on the serial monitor as functions are run. This way I can more easily see where something is not functioning correctly within the program. The first phase of the project was the creation of a “kill switch” function that would run when the temperature sensor hit a certain threshold. I selected the threshold of 75 degrees Celsius, as this is the maximum recommended operating temperature for the L293D. I believe phase 2 functions as intended, but it is difficult to determine with the visual glitches presented in TinkerCad. This has proven to be the biggest thorn in my side throughout the duration of the project. Phase three functions as intended, when the robot’s kill switch is triggered, it backs up, turns, and continues on its way.

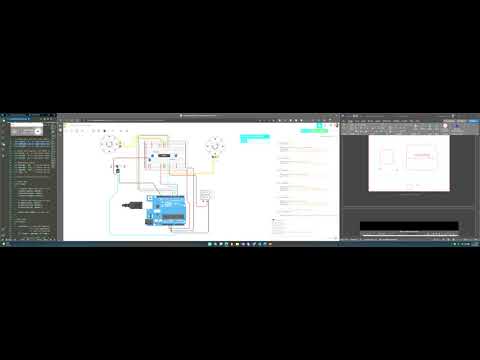
**Implementation**



FINAL PROJECT CIRCUIT



FINAL PROJECT DIAGRAM

[](https://www.youtube.com/embed/OTklqxa-0MA?feature=oembed)

FINAL PROJECT VIDEO

*// Final Project - Semi-Autonomous Robot*

*// Brandon Ackerman, RBT173*

*// GLOBAL VARIABLES*

*// Motor Control*

*// define pins used for right motor control*

*int* *rightEn* *=* 3; *// right motor enable*

*int* *forRight* *=* 4; *// right motor forwards*

*int* *revRight* *=* 5; *// right motor reverse*

*// define pins used for left Motor Control*

*int* *leftEn* *=* 6; *// left motor enable*

*int* *forLeft* *=* 7; *// left motor forwards*

*int* *revLeft* *=* 8; *// left motor reverse*

*// Temperature Sensor*

*int* *pinTemp* *=* A1; *// assign analog 1 as the temp sensor*

*int* *maxTemp* *=* (75); *// define the maximum cutoff temp for shutdown*

*int* *appTemp* *=* (65); *// define the temp at which the motors will begin slowing down*

*int* *temp* *=* 0; *// create an integer to store temperature value*

*////////////////////////////////////////*

*// Setup Code*

*void* *setup*()

{

*// define right motor pins as output*

*pinMode*(*rightEn*, OUTPUT);

*pinMode*(*forRight*, OUTPUT);

*pinMode*(*revRight*, OUTPUT);

*// define left motor pins as output*

*pinMode*(*leftEn*, OUTPUT);

*pinMode*(*forLeft*, OUTPUT);

*pinMode*(*revLeft*, OUTPUT);

*Serial*.*begin*(9600); *// Start the serial monitor*

}

*// Main Code*

*void* *loop*()

{

*readTemp*(); *// read the temperature*

*// if the temp is greater than or equal to the minimum start slowdown threshold,*

*//  and less than the maximum threshold,*

*if* ((*temp* *>=* *appTemp*) *&&* (*temp* *<* *maxTemp*))

    {

*analogWrite*(*rightEn*, 255 *-* 1); *// progressively slow down right motor using PWM*

*analogWrite*(*leftEn*, 255 *-* 1); *// progressively slow down left motor using PWM*

*Serial*.*println*("slowing down");

    }

*// if the temp reading is greater than or equal to the maximum threshold(75c)*

*if* (*temp* *>=* *maxTemp*)

*tooHot*(); *// run function tooHot()*

*else*

    {

*analogWrite*(*rightEn*, 255); *// send full speed PWM command to right motor to enable*

*analogWrite*(*leftEn*, 255); *// send full speed PWM command to right motor to enable*

*forward*(); *// run the forward function*

*delay*(500); *// wait 500ms*

    }

}

*// MOTOR SUB-FUNCTIONS*

*// Right Motor Forward*

*void* *rightFor*()

{

*digitalWrite*(*revRight*, LOW); *// set arduino pin 5 to LOW - denies power to L293 input2*

*digitalWrite*(*forRight*, HIGH); *// set arduino pin 4 to HIGH - applies power to L293 input1*

}

*// Right Motor Reverse*

*void* *rightRev*()

{

*digitalWrite*(*revRight*, HIGH); *// set arduino pin 5 to HIGH - applies power to L293 input2*

*digitalWrite*(*forRight*, LOW); *// set arduino pin 4 to LOW - denies power to L293 input1*

}

*// Left Motor Forward*

*void* *leftFor*()

{

*digitalWrite*(*revLeft*, LOW); *// set arduino pin 8 to LOW - denies power to L293 input4*

*digitalWrite*(*forLeft*, HIGH); *// set arduino pin 7 to HIGH - applies power to L293 input3*

}

*// Left Motor Reverse*

*void* *leftRev*()

{

*digitalWrite*(*revLeft*, HIGH); *// set arduino pin 8 to HIGH - applies power to L293 input4*

*digitalWrite*(*forLeft*, LOW); *// set arduino pin 7 to LOW - denies power to L293 input3*

}

*// MAIN FUNCTIONS*

*//  Drive Forward*

*void* *forward*()

{

*Serial*.*println*("driving forward");

*rightFor*(); *// run the right motor forward function*

*leftFor*(); *// run the left motor forward function*

}

*// Drive Backwards*

*void* *reverse*()

{

*Serial*.*println*("driving backwards");

*rightRev*(); *// run the right motor reverse function*

*leftRev*(); *// run the left motor reverse function*

}

*// Turn Right*

*void* *rightTurn*()

{

*Serial*.*println*("turning right");

*leftFor*(); *// run the left motor forward function*

}

*// Turn Left*

*void* *leftTurn*()

{

*Serial*.*println*("turning left");

*rightFor*(); *// run the right motor forward function*

}

*// Spin Left*

*void* *leftSpin*()

{

*Serial*.*println*("spin left");

*rightFor*(); *// run the right motor forward function*

*leftRev*(); *// run the left motor reverse function*

}

*// Spin Right*

*void* *rightSpin*()

{

*Serial*.*println*("spin right");

*leftFor*(); *// run the left motor forward function*

*rightRev*(); *// run the right motor reverse function*

}

*// function to immediately stop the car*

*void* *killSwitch*()

{

*Serial*.*println*("Maximum temperature threshold reached, stopping motors.");

*analogWrite*(*rightEn*, 0); *// disable right motor*

*analogWrite*(*leftEn*, 0); *// disable left motor*

}

*// Function to read the temperature*

*void* *readTemp*()

{

*Serial*.*println*("reading temperature");

*temp* *=* *analogRead*(*pinTemp*); *// Read the analog pin*

*temp* *=* (*temp* *\** 5.0 *\** 100.0) */* 1024.0; *// convert output (mv) to readable Celsius*

*// print the temperature status*

*Serial*.*print*("Temperature: ");

*Serial*.*print*(*temp*);

*Serial*.*println*("C");

}

*// function to run when upper temperature limit is reached*

*void* *tooHot*()

{

*killSwitch*(); *// run the killswitch function*

*delay*(2000); *// wait 2 seconds*

    {

*analogWrite*(*rightEn*, 128); *// enable right motor 1/2 speed*

*analogWrite*(*leftEn*, 128); *// enable left motor 1/2 speed*

*reverse*(); *// run the reverse function*

*delay*(5000); *// wait 5 seconds (reverse for 5 seconds)*

*rightTurn*(); *// run the right turn function*

*delay*(2000); *// wait 2 seconds (turn for 2 seconds)*

    }

}