Project 1 Content

- A simple hands-on example: blink an onboard LED
- Arduino programming Introduction
- Hands-on 1: Blink LEDs
- Hands-on 2: Control LEDs
- Hands-on 3: Ultrasonic Sensor
- Hands-on 4: Vehicle Controls

Ultrasonic Sensor

- To experiment with reading a digital value, we will use HC-SR04, an ultrasonic ranging sensor.
- It provides 2 cm – 4 m non-contact distance measurement function.
- The ranging accuracy can reach to 3 mm.

Ultrasonic (cont.)

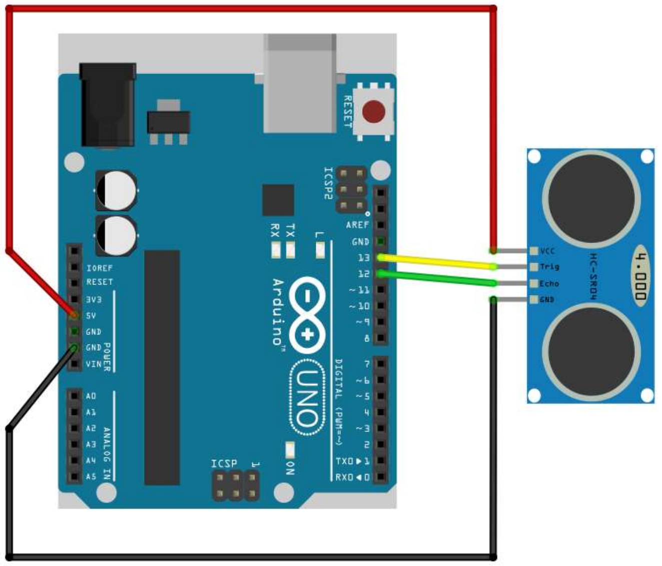
Task: Read one ultrasonic sensor distance data.

❖ You need the following:

❖ You can also use a breadboard.

Ultrasound: Connection

Task: Read one ultrasonic sensor distance data.



Ultrasound: Programming

```
#define trigPin 13   //Trig to Arduino pin 13
#define echoPin 12   //Echo to Arduino pin 12

void setup() {
  Serial.begin (9600);        //Serial communications at 9600 bps
  pinMode(trigPin, OUTPUT);   //Set the trigPin as output
  pinMode(echoPin, INPUT);    //Set the echoPin as input
}

void loop() {
  float duration, distance;

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
```

Ultrasound: Programming

```
  duration = pulseIn(echoPin, HIGH); //Read the HIGH pulse whose
duration is the time (in microseconds) from the sending of the
ping to the reception of its echo off of an object.

  distance = duration; // The distance should be a function of
duration, which needs a calibration from data.

  //Prints data to the serial port as human-readable ASCII text.
  Serial.print(distance);
  Serial.println(" mm");

  delay(100);
}
```

Ultrasound: Programming

```
#define trigPin 13   //Trig to Arduino pin 13
#define echoPin 12   //Echo to Arduino pin 12
void setup() {
  Serial.begin (9600);        //Serial communications at 9600 bps
  pinMode(trigPin, OUTPUT);   //Set the trigPin as output
  pinMode(echoPin, INPUT);    //Set the echoPin as input
}

void loop() {
  float duration, distance;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = duration; //Function TBD
Serial.print(distance);
Serial.println(" mm");
delay(100);
}
```

Upload and Run the Arduino Code

❖ Save the code as *ultrasound1*.

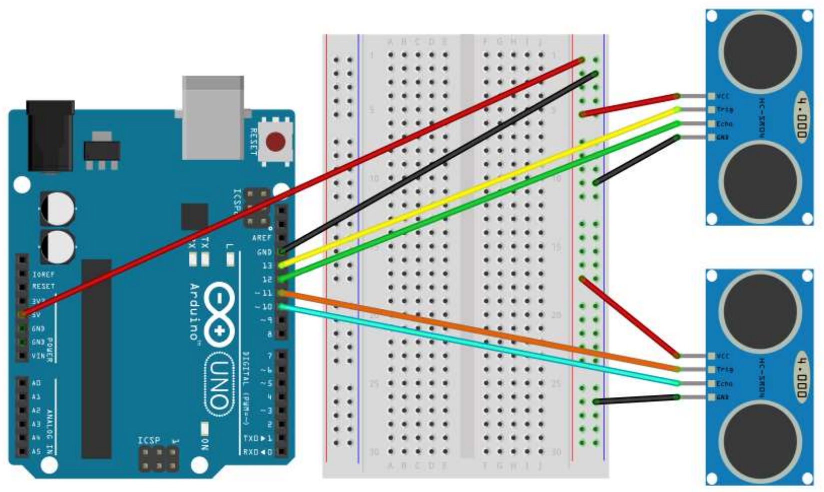
❖ Connect the board and upload the code.

❖ Open the serial monitor by clicking on

❖ Experiment by placing an object closer and further from the sensor.

Challenge: Read 2 Ultrasonic Sensors

Task: Read 2 ultrasonic sensor distance data.



Read 2 Ultrasonic Sensors

```
#define trigPin1 13
#define echoPin1 12
#define trigPin2 11
#define echoPin2 10
void setup() {
  Serial.begin (9600);       //Serial communications at 9600 bps
  pinMode(trigPin1, OUTPUT);  //Set the trigPin1 as output
  pinMode(echoPin1, INPUT);   //Set the echoPin1 as input
  pinMode(trigPin2, OUTPUT);  //Set the trigPin2 as output
  pinMode(echoPin2, INPUT);   //Set the echoPin2 as input
}
void loop() {
  float duration1, distance1, duration2, distance2;
  digitalWrite(trigPin1, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin1, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin1, LOW);
```

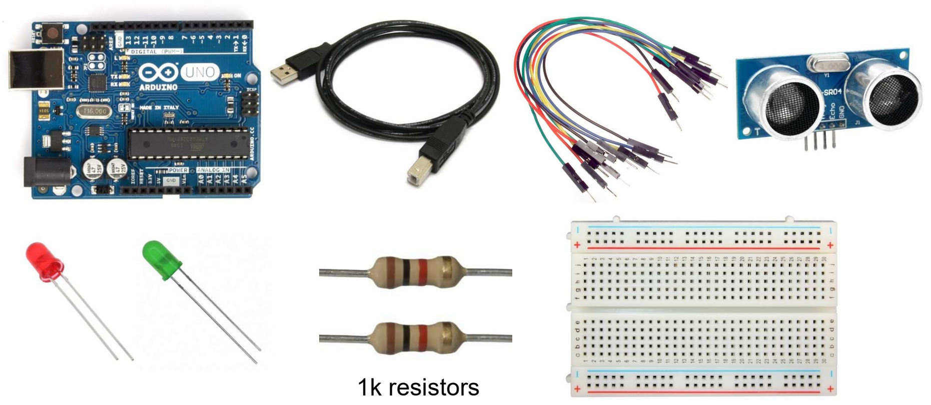
Read 2 Ultrasonic sensors

```
  duration1 = pulseIn(echoPin1, HIGH, 2000); // Arduino will only wait
2000 microseconds for the pulse to be completed.
  distance1 = duration1; // Function TBD
  Serial.print(distance1);
  Serial.print(" mm  ");
  delayMicroseconds(20);

  digitalWrite(trigPin2, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin2, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin2, LOW);
  duration2 = pulseIn(echoPin2, HIGH, 2000); // Arduino will only wait
2000 microseconds for the pulse to be completed.
  distance2 = duration2; // Function TBD
  Serial.print(distance2);
  Serial.println(" mm");
  delay(100);
}
```

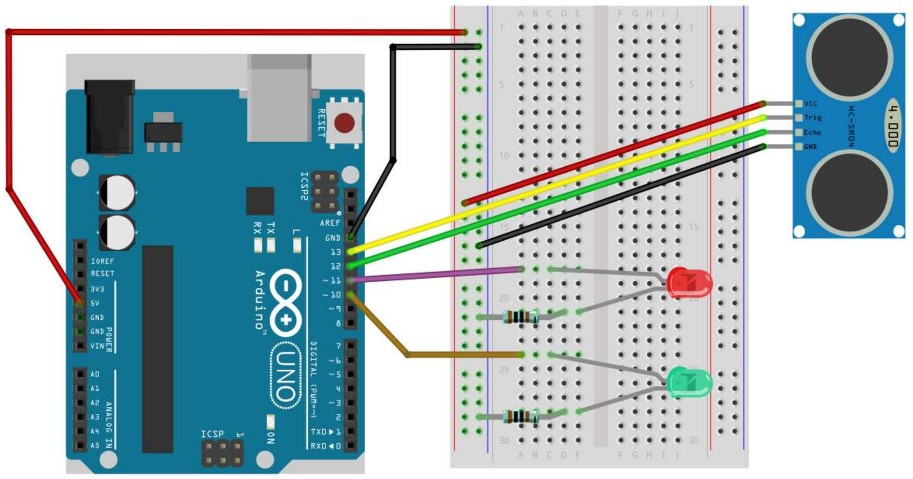
Challenge: Proximity Warning

Task: Use an ultrasonic sensor to measure the distance to an object. Turn on the red LED if the object is less than 20 cm away from the sensor. Otherwise, turn on the green LED.

❖ You need the following:



Proximity Warning: Connection



CLEMSON UNIVERSITY INTERNATIONAL CENTER FOR AUTOMOTIVE RESEARCH

66

Proximity Warning: Programming

```
#define trigPin 13   //Trig to Arduino pin 13
#define echoPin 12   //Echo to Arduino pin 12
#define redLed 11    //Red positive to Arduino pin 11
#define greenLed 10  //Green positive to Arduino pin 10

void setup() {
  Serial.begin (9600);        //Serial communications at 9600 bps
  pinMode(trigPin, OUTPUT);   //Set the trigPin as output
  pinMode(echoPin, INPUT);    //Set the echoPin as input
  pinMode(redLed, OUTPUT);    //Set the redLed as output
  pinMode(greenLed, OUTPUT);  //Set the greenLed as output
}
```

CLEMSON UNIVERSITY INTERNATIONAL CENTER FOR AUTOMOTIVE RESEARCH

67

Proximity Warning: Programming

```
void loop() {
  float duration, distance;

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH); //  Read the HIGH pulse
whose duration is the time (in microseconds) from the sending of
the ping to the reception of its echo off of an object.

  distance = duration; //Function TBD
```

Proximity Warning: Programming

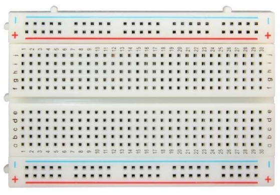
```
 // This is where the LED On/Off happens
  if (distance < 20) {
    digitalWrite(redLed,HIGH);
    digitalWrite(greenLed,LOW);
  } else {
    digitalWrite(redLed,LOW);
    digitalWrite(greenLed,HIGH);
  }
  if (distance >= 200 || distance <= 0){
    Serial.println("Out of range");
  }
  else {
    Serial.print(distance);
    Serial.println(" cm");
  }
  delay(100);
}
```

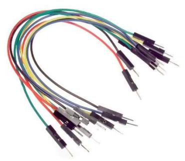
Project 1 Content

- A simple hands-on example: blink an onboard LED
- Arduino programming Introduction
- Hands-on 1: Blink LEDs
- Hands-on 2: Control LEDs
- Hands-on 3: Ultrasonic Sensor
- Hands-on 4: Vehicle Controls

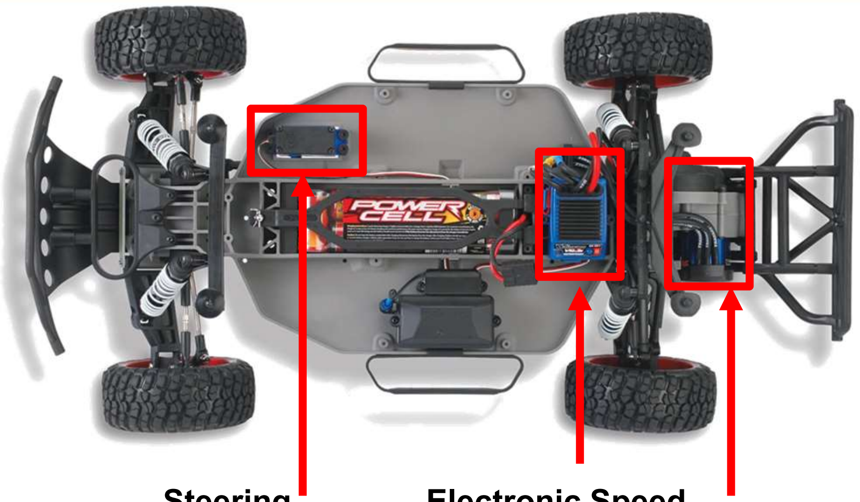
Racing Car for the Project 2

❖ You need the following





The Vehicle



Steering servo motor

Electronic Speed Control (ESC)

Motor

CUICAR CLEMSON UNIVERSITY INTERNATIONAL CENTER FOR AUTOMOTIVE RESEARCH

72

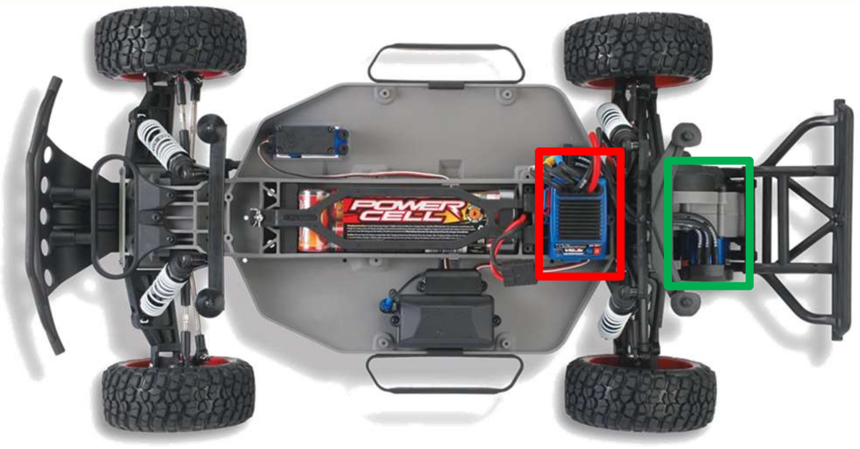
Steering Servo Motor

Steering servo: This controls the steering of the front wheels of the vehicle. It requires 5V input.

CUICAR CLEMSON UNIVERSITY INTERNATIONAL CENTER FOR AUTOMOTIVE RESEARCH

73

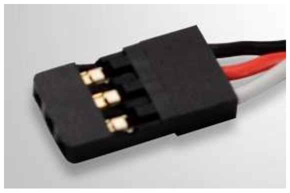
Electronic Speed Controller (ESC)



ESC: This controls the speed of the motor. It also produces 5V output for servo.

ESC and Servo Connections

- The ends of ESC and servo cables look like this:
  - We have 1 of these for each actuator



- We want to make connections that allows Arduino to control steering servo (steering) and ESC (speed) of the vehicle

Remember:

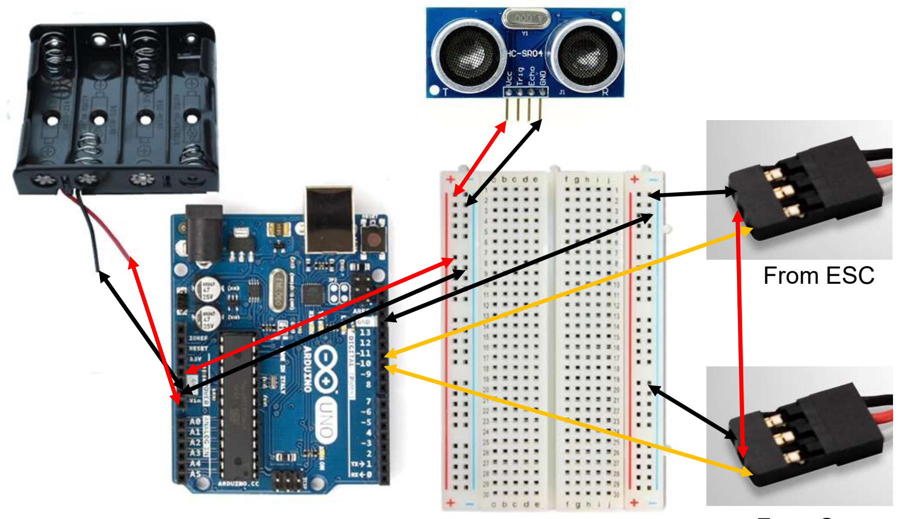
White – Signal
Red – Voltage (5V)
Black – Ground

Arduino's Power Supply

- ❖ To stabilize the readings from the sensors, you'll need a separate power system for sensors and Arduino.
- ❖ Use battery holder (4XAA batteries, 6V output) as the independent power source.
- ❖ Create circuit so that:
  - ESC supplies power to servo motor
  - Arduino, ESC, Servo, and battery pack have common ground
  - Connect the battery pack to the Vin port and GND port of Arduino
  - Connect the 5v output port of Arduino to the VCC ports of sensors

CLEMSON UNIVERSITY INTERNATIONAL CENTER FOR AUTOMOTIVE RESEARCH

Wiring



From ESC

From Servo

CLEMSON UNIVERSITY INTERNATIONAL CENTER FOR AUTOMOTIVE RESEARCH

Laptop safety

- **Do Not** connect laptop to Arduino while Arduino is supplied with battery pack.
- To observe the vehicle input on serial monitor, you need to connect your PC to the Arduino. So if you plug Arduino into your PC as well as connect Arduino to battery pack's power with VIN, there may be damage to your laptop.
- So if you want to upload the codes or observe serial output while the vehicle is powered on, you have to remove the *Vin* line from your Arduino. Failing to do so may end you up with a dead USB port and maybe even a dead laptop.

CLEMSON UNIVERSITY INTERNATIONAL CENTER FOR AUTOMOTIVE RESEARCH

Servo control with Arduino

- Arduino has a dedicated library for servo motors, called *Servo.h*. Usually, servos can be controlled between 0 and 180 here, with default being 90.
- So for the vehicle, 90 would be the default where it does not steer and or move. Setting ESC at 100, it will move forward and at 80 it will move backward. Similarly, setting steering to 45 will turn left and 135 will turn right.

CLEMSON UNIVERSITY INTERNATIONAL CENTER FOR AUTOMOTIVE RESEARCH

Vehicle control code – Initial part

Download basecode.ino from Canvas. Below is the code:

```
#include <Servo.h> //define the servo library

Servo ssm; //create an instance of the servo object called ssm
Servo esc; //create an instance of the servo object called esc

int steering=90,velocity=90; //defining global variables to use later
```

Vehicle control code – Setup part

```
void setup() {
  //Serial.begin(9600); //start serial connection. Uncomment for PC

  ssm.attach(10);       //define that ssm is connected at pin 10

  esc.attach(11);       //define that esc is connected at pin 11
}
```

Vehicle control code – Loop code

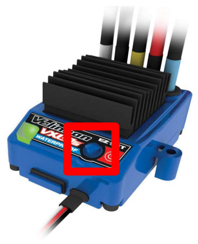
```
void loop() {

// Add control logic here
steering=90;
velocity=90;

// Call the setVehicle function to set the vehicle steering and
velocity(speed) values
setVehicle(steering, velocity);
}
```

Vehicle control code – setVehicle function

```
//********************** Vehicle Control *********************//
//**************** Do not change below part ****************//
void setVehicle(int s, int v)
{
  s=min(max(0,s),180);  //saturate steering command
  v=min(max(75,v),105); //saturate velocity command
  ssm.write(s); //write steering value to steering servo
  esc.write(v); //write velocity value to the ESC unit
}
//**************** Do not change above part ****************//
```

Start and Stop the Vehicle

- After the code and wires are connected properly, you can start the vehicle.
- To start the vehicle, press the EZ-SET button on ESC.
- Do not hold for too long as it will enter calibration mode after holding the button for 2 seconds
- When the vehicle is properly started, the led light will first blink green, and then turn red.



Start and Stop the Vehicle

- To stop the vehicle, press and hold the EZ-SET button on ESC for 5 second. The ESC will be turned off when you release the button. Remember to detach Arduino Vin Pin.
- The *setVehicle function* is used to send velocity and steering signals to the vehicle, and limit the vehicle's speed in a safe range. *Do not modify this part*.

Practice: Speed Control

- Task: Download the base code from Canvas titled “basecode.ino”. Create logic for the below challenges.

  Challenge 1: Control the speed
  - Make sure the vehicle’s rear wheels do not touch the ground while uploading the code.
  - Drive forward by setting velocity to 100, keep 2 seconds.
  - Brake the vehicle by setting velocity to 90, keep 2 seconds.
  - Drive back by setting velocity to 80, keep 2 seconds.
  - Brake the vehicle by setting velocity to 90, keep 2 seconds.
  - Repeat the steps above.
  - *Notice: Without braking, some vehicles may not be able to reverse, and sometimes even the motor will be stalled.*

86

Switch between driving Forward and Backward

- Change from forward to reverse

$$\begin{tikzcd}
\boxed{\begin{array}{c}\text{Forward}\\ \text{e.g. 101}\end{array}} \arrow[r] & \boxed{\begin{array}{c}\text{Brake}\\ \text{e.g. 85}\end{array}} \arrow[r] & \boxed{\begin{array}{c}\text{Neutral}\\ \text{90}\end{array}} \arrow[r] & \boxed{\begin{array}{c}\text{Reverse}\\ \text{e.g. 80}\end{array}}
\end{tikzcd}$$

87

Example: Speed Control

- Add this code to basecode.ino

```
void loop() {
  setVehicle(90, 100);
  delay(2000);
  setVehicle(90, 90);
  delay(2000);
  setVehicle(90, 80);
  delay(2000);
  setVehicle(90, 90);
  delay(2000);
}
```

Practice: Steering Control

- Task: Download the base code from Canvas titled “basecode.ino”. Create logic for the below challenges.

Challenge 2: Control the steering

- Make sure the vehicle’s rear wheels do not touch the ground.
- Steer to left by setting steering to 45, keep 2 seconds.
- Steer back to neutral by setting steering to 90, keep 2 seconds.
- Steer to right by setting steering to 135, keep 2 seconds.
- Steer back to neutral by setting steering to 90, keep 2 seconds.
- Repeat the steps above.

Example: Steering Control

- Add this code to basecode.ino

```
void loop() {
  setVehicle(45, 90);
  delay(2000);
  setVehicle(90, 90);
  delay(2000);
  setVehicle(135, 90);
  delay(2000);
  setVehicle(90, 90);
  delay(2000);
}
```

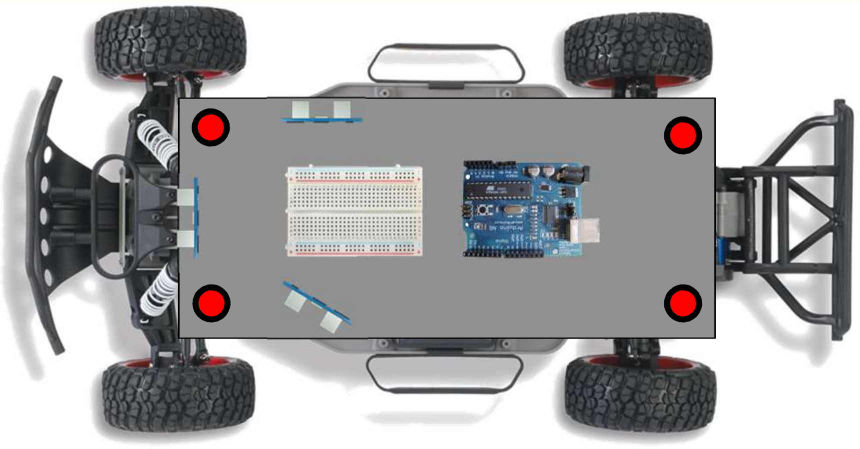
90

Building the fixture

- Design your configuration of ultrasonic sensors to achieve the adaptive cruise control and lane keeping functions in Project 1
- Design your controllers for speed control and steering control based on ultrasonic sensing (you can use any feedback controllers)
- To setup your own design, you'll have to build a fixture that you can use to mount your Arduino, breadboard and the ultrasonic sensors.
- You need to make sure your design is safe and won't damage any equipment.

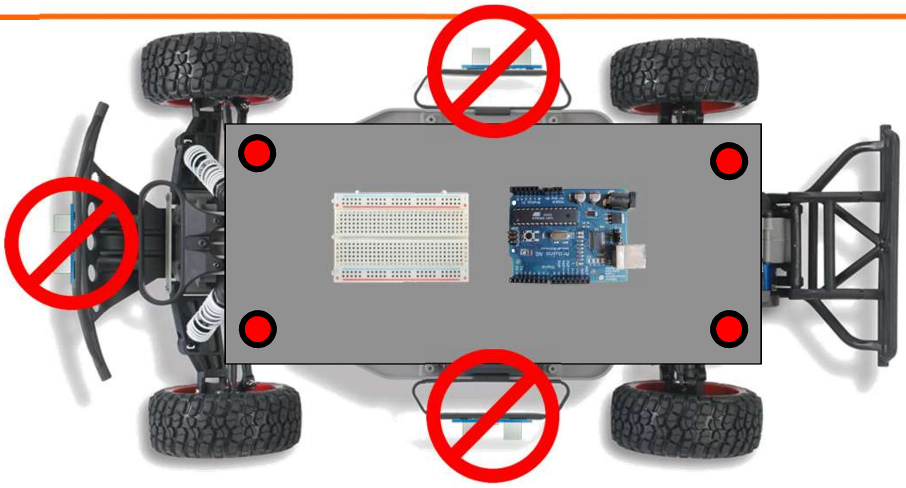
91

Setting up the RC car



One example of setup

Setting up the RC car



Do not place the sensors at the edge of the RC car. You'll likely end up damaging your sensors while testing the RC car.