```
# define ButtonPin1 12
# define ButtonPin2 3
# define ForeLED 13      // the number of the LED pin
# define revLED 8      // the number of the LED pin
# define ledPin 4      // the number of the LED pin

# define Trig 6
# define Echo 5

//Button 1 for the foreward/reverse
bool button1State;
bool lastButton1State;
long lastDebounceTime1 = 0;

//button2 for the whole frame
bool button2State;
bool lastButton2State;
long lastDebounceTime2 = 0;

//State of the motor and direction
bool revState=LOW;
bool foreState=HIGH;
bool motorState=LOW;
bool ledState = HIGH;
const long debounceDelay = 50;


int ForewardA =10 ;// Input 1
int ReverseA =11 ;  // Input 2
int motorPin = 9;   // Used to send commands to the buggy using PWM
int pwmspeed = 0;
const int maxspeed = 225; //between 0 and 255

long duration = 1000;
int Ultradistance;            // variable for storing the distance the
buggy is from an object using the ultrasonics.
const int distance = 25;

void setup(){           // void setup is only run once at the very
beginning.
    // ULTRASONICS

pinMode(ButtonPin1,INPUT);
pinMode(ButtonPin2,INPUT);

pinMode(ledPin,OUTPUT);
pinMode(revLED,OUTPUT);
pinMode(ForeLED,OUTPUT);

pinMode(ForewardA,OUTPUT);
pinMode(ReverseA,OUTPUT);
digitalWrite(ReverseA, revState);
digitalWrite(ForewardA, foreState);

  pinMode(Trig, OUTPUT);
  pinMode(Echo, INPUT);
  pinMode(1, OUTPUT);
      // MOTOR (PWM)
```

```
  pinMode(motorPin, OUTPUT);

Serial.begin(9600);
}
void loop()
{

  Button1funt();
  Button2funt();
// ULTRASONICS
if(motorState==HIGH){
      Ultrasonics();
}
else
 analogWrite(motorPin, 0);


}


void Button1funt(){
  boolean reading = digitalRead(ButtonPin1);
//Serial.println(reading);
  if (reading != lastButton1State) {
    lastDebounceTime1 = millis();

  }
  if ((millis() - lastDebounceTime1) > debounceDelay) {
    if (reading != button1State) {
      button1State = reading;
      if (button1State == HIGH) {
        //Serial.println("Blahhh");
        revState =!revState;
        foreState=!foreState;
        digitalWrite(ReverseA, revState);
        digitalWrite(ForeLED, foreState);
        digitalWrite(revLED, revState);
        digitalWrite(ForewardA, foreState);
}
    }
  }
    lastButton1State = reading;

}

void Button2funt(){
  boolean reading = digitalRead(ButtonPin2);
  if (reading != lastButton2State) {
    lastDebounceTime2 = millis();
  }

  if ((millis() - lastDebounceTime2) > debounceDelay) {
    if (reading != button2State) {
      button2State = reading;
      if (button2State == HIGH) {
             //   Serial.println("Blahhh22222222222222");

        motorState = !motorState;
```

```
                digitalWrite(ledPin, ledState);
          ledState = !ledState;

        }
      }
    }
    // save the reading.  Next time through the loop,
    // it'll be the lastButtonState:
    lastButton2State = reading;
  }

  void Ultrasonics(){
    //if(Ultradistance!=DistanceMeasure()){
     Ultradistance=DistanceMeasure();    //calls the function and stores it
     delay(10);
     //Serial.println(Ultradistance);
     //Serial.println(Ultradistance);
      // Button2funt();
       //Button1funt();


     if (Ultradistance<=distance) {
        pwmspeed+=30;
        if (pwmspeed>=maxspeed)
       pwmspeed=maxspeed;
        Serial.print("The distance between the user and the frame is: ");
      Serial.print(Ultradistance);
     Serial.print(" cm. The arduino will increase the speed to: ");
      Serial.print(pwmspeed);
       Serial.println(" pwm");
        }
     else if(Ultradistance>=distance){
        pwmspeed-=40;
        if(pwmspeed<=0)
       pwmspeed=0;
        Serial.print("The distance between the user and the frame is: ");
        Serial.print(Ultradistance);
        Serial.print(" cm. The arduino will decrease the speed to: ");
        Serial.print(pwmspeed);
       Serial.println(" pwm");
      // Button1funt();
      //  Button2funt();

     }

   // }
    analogWrite(motorPin, pwmspeed);

  }



   long DistanceMeasure(void)    // used for sending an Ultra sonic singal
  and then seeing how long it takes to come back to the buggy.
                                  // This then can calculate the how far the
  buggy is from the obstacle and return that value.
  {
    digitalWrite(Trig, LOW);
```

```
175     delayMicroseconds(2);
176
177     digitalWrite(Trig, HIGH);
178     delayMicroseconds(10);    // sends a signal for at least 10 Micros.
179
180     digitalWrite(Trig,LOW);
181
182     duration = pulseIn(Echo,HIGH);// measures how long it takes for the
183   signal to get back to the buggy.
184   //Serial.println(duration/58.2);
185     duration = ((duration/2)*340*0.000001*100); // divides that time in two
186   to get the time from the buggy to the abstacle.
187                                                 // multiplies it by the
188   speed of sound (340 m/s) then *10^-6 to convert from micro secounds to
189   secounds.
190
191
192     return duration;          // returns the distance. in cm
193   }
```