Colorado School of Mines

Memo

To: Professor Xiaoli Zhang [Songpo Li]

From: Sam Mcalexander and Ian Cairns

Team #: 413

Date: 12/02/15

Re: Lab 6

Problem Statement:

The objective for this lab was to create a robot similar to a mining robot in which the robot must locate and retrieve an item. A remote control is used to control the robot, up to where the robot enters the “mine” (a tunel similar to our mazes). At that point the robot takes over in order to simulate a situation in which communication between the robot and the remote control is lost. The robot effectively simulates the effect of a tunnel, where the object that the robot has to retrieve is located at the end of the tunnel. Therefore, the automated control system must be capable of retrieving the object and then backing the robot out of the tunnel. After exiting the tunnel the robot allows the remote control to take over. A 3D printed robotic arm is used to retrieve the item and is also able to be controlled by the remote.

Methods:

The overall design of the robot utilizes two Arduino microcontrollers, two IR sensors, a remote control, a remote control receiver, and a 3D printed robotic arm using 3 servos. The left IR sensor is shown in figure 2, whereas the front IR sensor is shown in figure 3. The 3D printed robotic arm shown in figure 4, is attached as shown in figure 5. The remote control and receiver are shown in figure 6 and 7 respectively. The Arduino Uno controls the motor, while the Arduino Nano controls the arm.

Since two Arduinos were used, the wiring was more complex than normal. Wiring for the IR sensors is shown in figure 8. The receiver is wired to pins 4 and 5 on the Arduino Uno as shown in figure 9 for channels 1 and 2 in order to control the car. In order to control the robotic arm the receiver is wired to pins 4 and 3 on the Arduino Nano for channels 1 and 2 as shown in figure 10. Pins 6 and 8 on the Arduino Uno are connected to pins 5 and 6 on the receiver as shown in figure 11. The servo control pins are connected to

the Arduino Nano on pins 11,10 and 9. Figure 12 illustrates how the servos are wired to the receiver.

In order to simulate the effect of a tunnel, the robot is conditionally coded to automatically take over, in which case the remote control cannot be used to command the robot. This was accomplished through the use of an IR sensor attached on the left of the car, as seen in figure 2. and a time duration. If the robot is within a certain distance of a wall to its left, for 3 seconds, it will automatically take over control of the car. Another IR sensor placed in the front of the car as seen in figure 3 is used to detect the front wall where the object will stop and pick up the object. Two PID loops are used in this project, one on the side walls in order to prevent collisions with the tunnel, and one on front wall where the robot will retrieve the object. The PID on the front wall stops the robot at an appropriate distance away from the object, and allows it to pick the object up by use of the robotic arm.

After the robot retrieves the object, it then backs out of the tunnel before the user can regain control of it. Once the robot is safely out of the tunnel, the user regains control and can then set the object down somewhere by use of the remote.

At the start of the Uno code it initializes the vehicle into the remote state for the car. The code then reads the remote using the remote function and then calculates the pwm based on the values it receives from the remote. The receiver from the remote outputs a signal that is meant for a servo. It is a for of pwm. I use the pulse in function to measure the duration of the on cycle of the pwm. I then map the time I get back, which is between 1000 and 2000, onto the output I desire for the motion of the car. In remote mode the car also is listening for the steer to be turned to the left for 3 seconds with no throttle. This changes the modes between the remote controlling the car vs. the arm. When the mode is flipped to the arm state the uno turns pin 6 high which is connected to the nano and tells the nano code that it can now start using the remote signal to control the arm. When the pin is turned back to low the same way it was turned high the arm nano stops controlling the arm. The nano reads the remote the same way the uno does. It uses the pulsein command to record the duration of the high portion of the receivers duty cycle. It then checks which way the throttle was going. If it is forward it controls the arm moving up and down. If it is backward it controls the arm by moving the gripper. It uses the turn portion of the remote to know which way to move the selected portion of the arm. There is an if statement around this portion of code that only allows it to run when the pin from the uno is set to high. During the Remote portion of the code the uno is also checking to see if there is something close to the left distance sensor. If so it records that and checks if the object is there for 3 seconds. If so it enters it autonomous state where it drives forward using a PID scheme on the left wall. When it detects the wall in front of it, it starts to run a PID scheme on

the front wall. It runs the PID around a point about 15 cm away from the wall. Once it reaches the wall it turns pin 8 high which is connected to the nano. This then tells the nano to run its autonomous route. The Uno then waits 25 seconds for this to happen. The Nano has a path array for picking up the object. When pin 8 from the uno goes high it runs this path. The path is stored as locations and the code iterates the position of the servos until it reaches the desired location of the array. After the nano finishes picking up the object the robot then backs out of the maze. It does this again by using a PID scheme on the left wall. It it finds that it has backed out and couldn't see the wall for 3 seconds it re-enters the remote mode where it allows the user to control the robot and drop of the object where ever they want.

Results:

The robot successfully navigated through the 'tunnel' and located the object it needed to retrieve. The best method for keeping the robot from running into the side of the tunnel was by incorporating PID on the wall. This allows the robot to recover if it is angled towards the wall, and prevents the robot from crashing. It was interesting getting the switch aspect between remote controlling arm and car to work properly. Once we got that working we were able to effectively switch between car and arm control. The next big obstacle that we encountered was in getting the robot to allow for the user to take back control, after its autonomous role. Another big problem we had was the mounting of the left sensor. It works best for PID when the sensor is in front of the center of rotation. When it is behind the sensor actually gets further away from the desired position when it turns to go towards the desired position. This leads to a somewhat out of control PID. We have never crashed but it looks strange. This project successfully simulates a real world scenario in which the robot can lose communication and must carry out the majority of its task without remote control support.

Conclusions:

Overall this was an interesting project. Both Arduinos had to be carefully programmed in order to ensure seamless switching for the controller. The main difficulties encountered in this project were primarily programming based. We had to calibrate the robot to stop at just the right distance away from the object, otherwise the robotic arm would not be able to retrieve it. The main assumption of this project is that the tunnel is essentially a straightaway, which means that this would not work if the tunnel was more like a maze. Incorporating a maze solving algorithm into this project was not within our scope and therefore was not included as a goal. This could be integrated for a more complex robot.

References

[1] Blackboard.mines.edu, 2015. [Online]. Available: https://blackboard.mines.edu/webapps/portal/frameset.jsp?tab_tab_group_id=_1_1&url=%2Fwebapps%2Fblackboard%2Fexecute%2Flauncher%3Ftype%3DCourse%26id%3D_26212_1%26url%3D. [Accessed: 09- Sep- 2015].

[2] “Arduino.”.Web.9 sept.2015. https://www.arduino.cc

[3] "Ardumoto Shield Hookup Guide." Ardumoto Shield Hookup Guide. Web. 9 Sept. 2015. <https://learn.sparkfun.com/tutorials/ardumoto-shield-hookup-guide>.

Appendix:

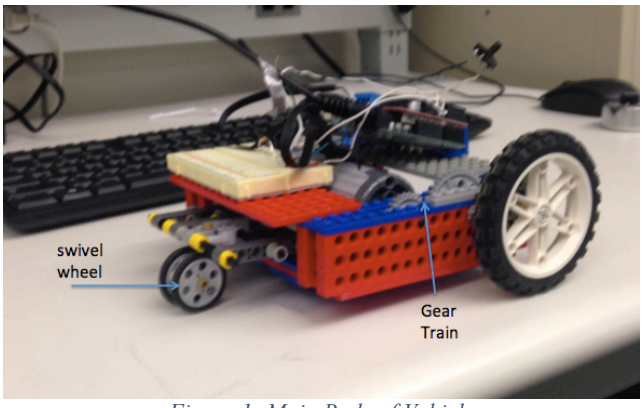


Figure 1: Main Body of Vehicle

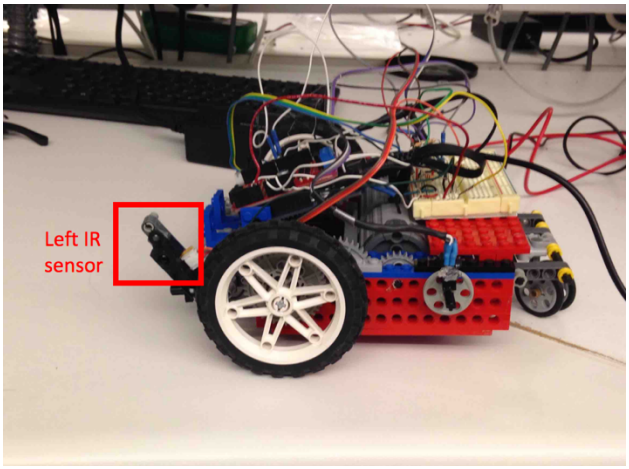


Figure 2: Left IR Sensor

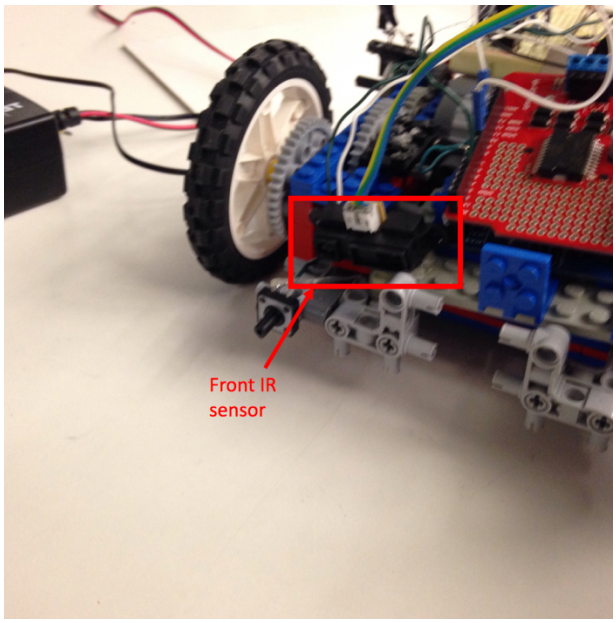


Figure 3: Front IR Sensor

Figure 4: Robotic Arm

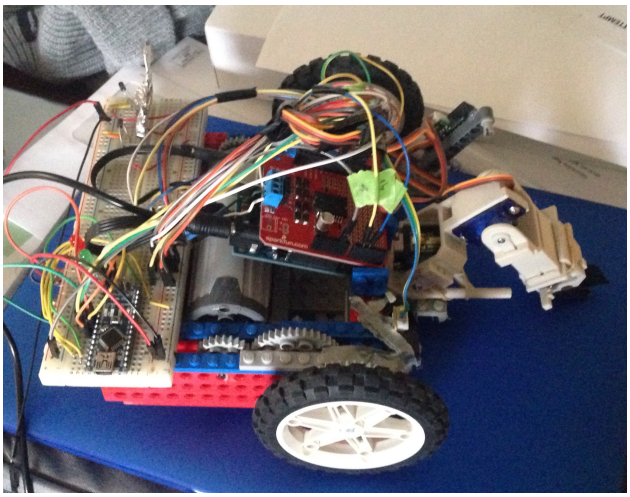


Figure 5: Robotic Arm on Car

Figure 6: Remote Control

Figure 7: Receiver

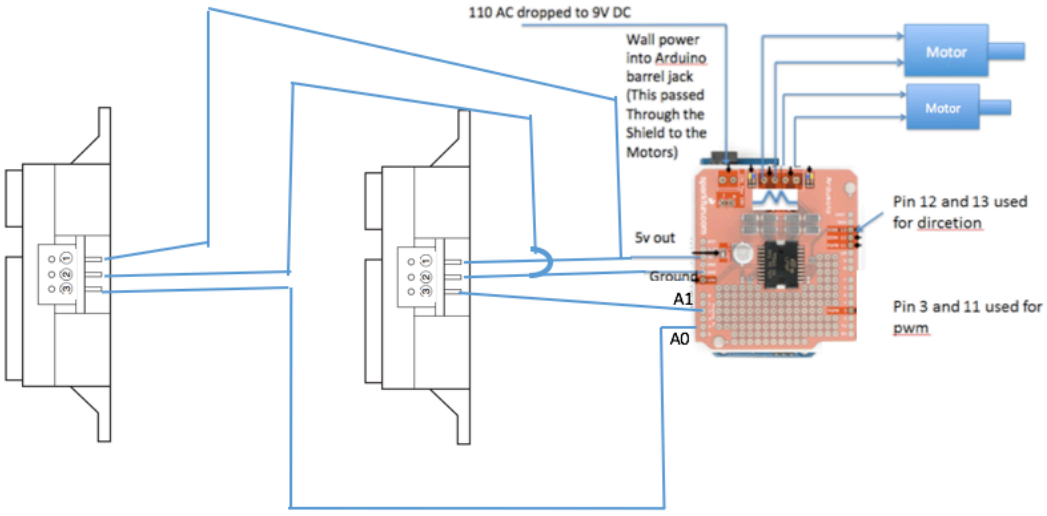


Figure 8: IR Sensor Wiring

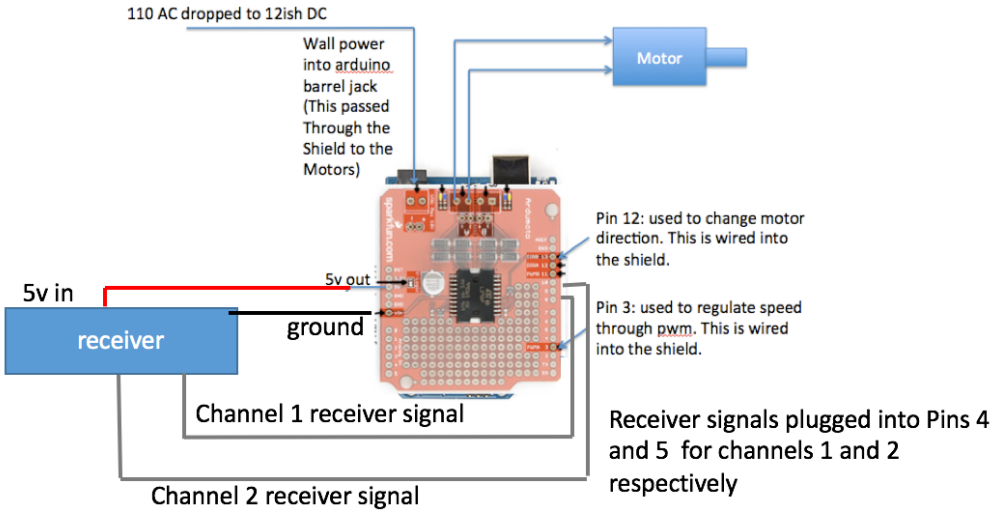


Figure 9: Receiver Wiring for Arduino Uno

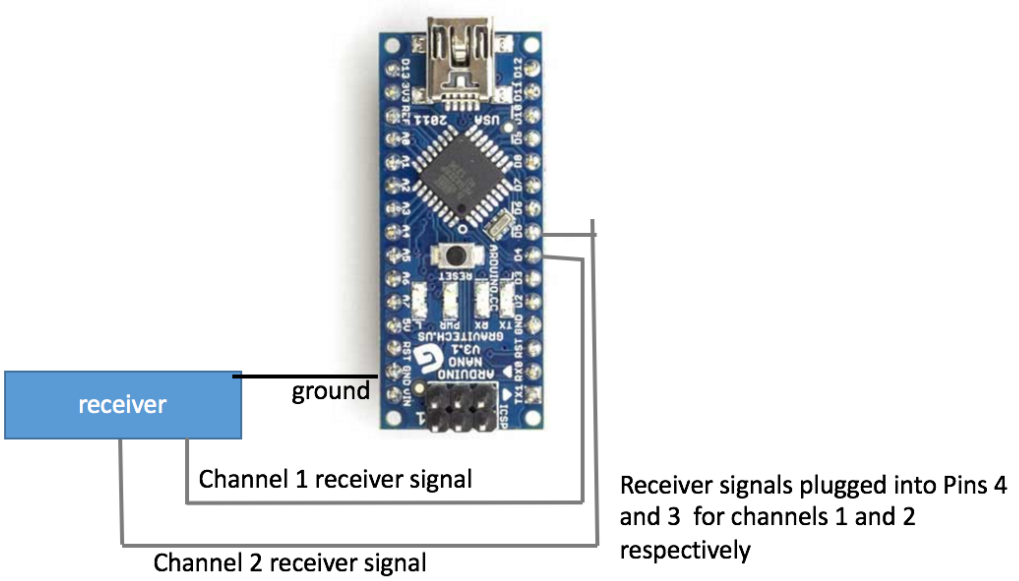


Figure 10: Receiver Wiring for Arduino Nano

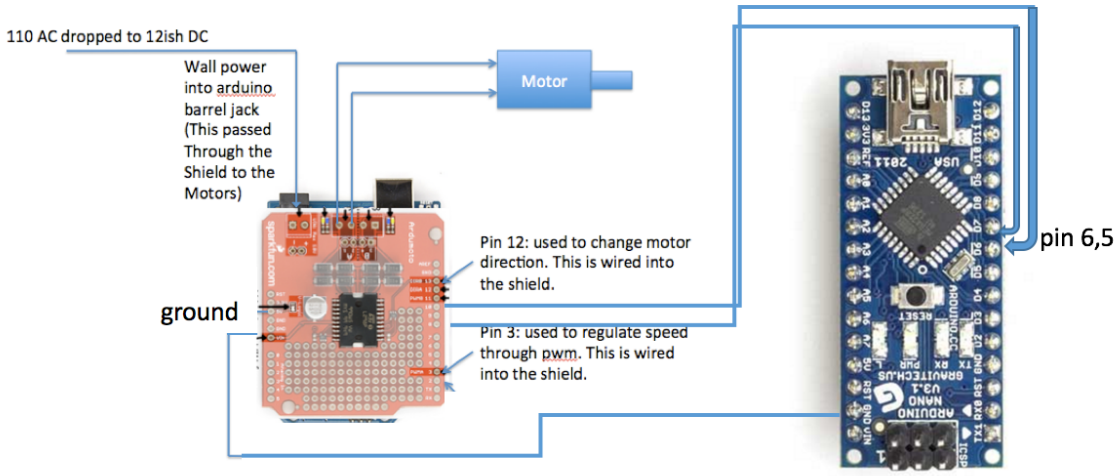
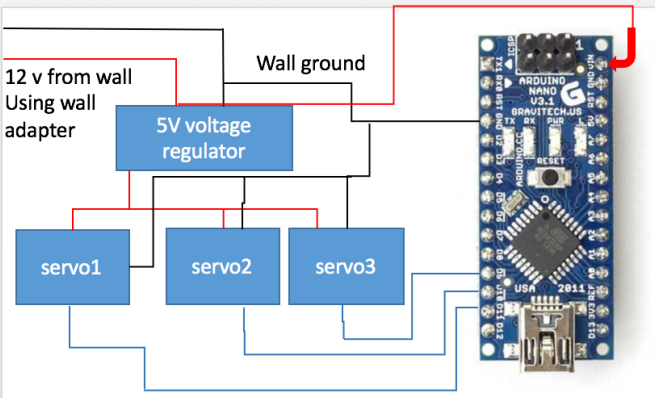


Figure 11: Co-grounded. Pin 10 and 6 on Main Arduino Signaling to Pin 6 & 7 Signaling to Arm



Servo control pins (require pwm input)
Servo control pins are plugged into pins 11,10,9 on the arduino nano.
Power for this arduino and the servos comes from a wall adaptor converted to plug into a
Bread board. There will be two power supplies used in this system. One with the barrel jack for the
Motor arduino and one for the second arduino and servos.

Figure 12: Servos Wiring Diagram

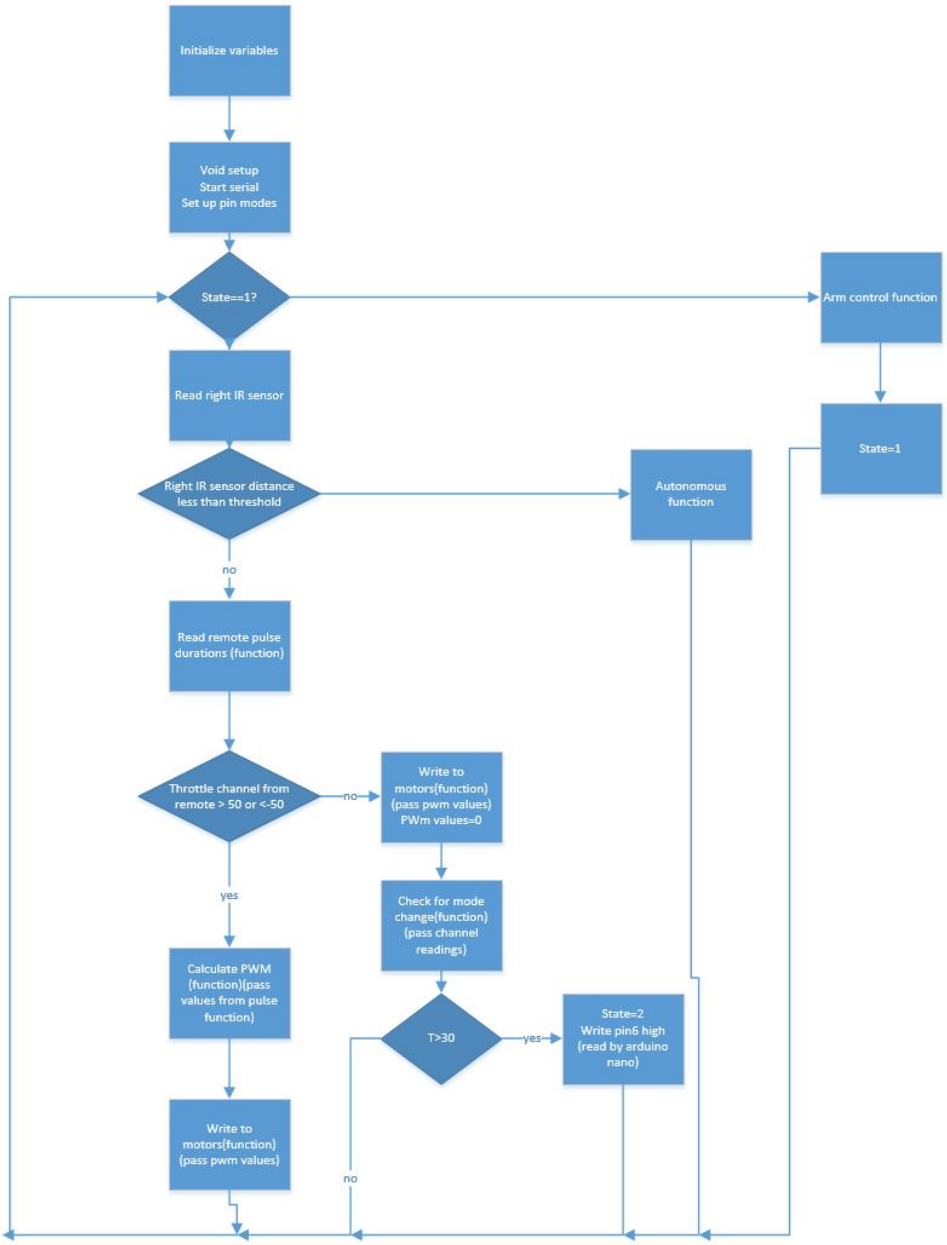


Figure 13: Driving Arduino Main Loop

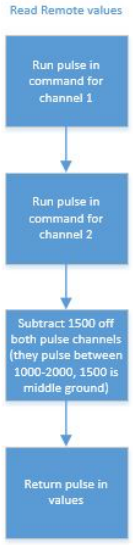


Figure 14: Read Remote Values Function

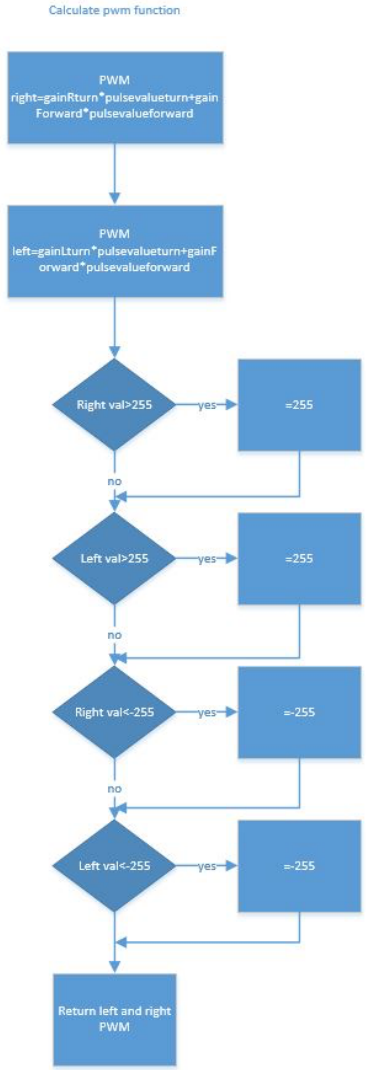


Figure 15: Calculate PWM Function

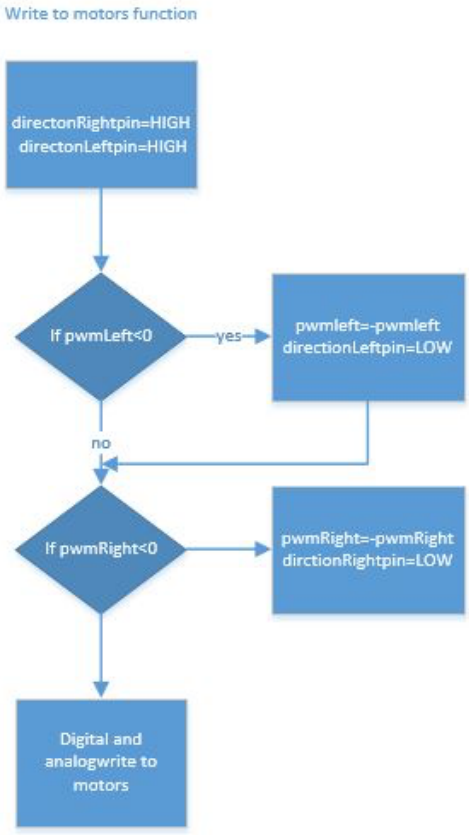


Figure 16: Motor Function

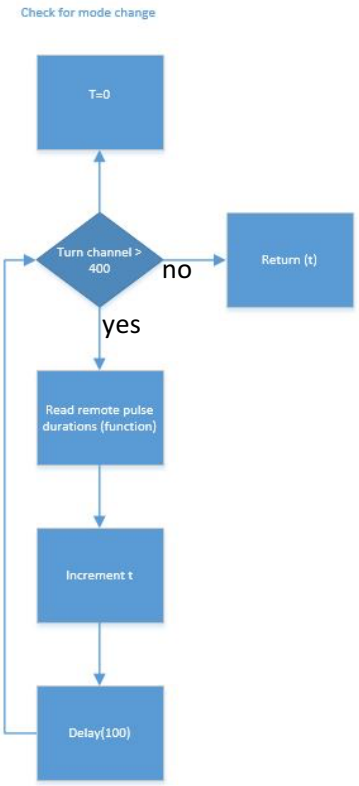


Figure 17: Check for Mode Change Function

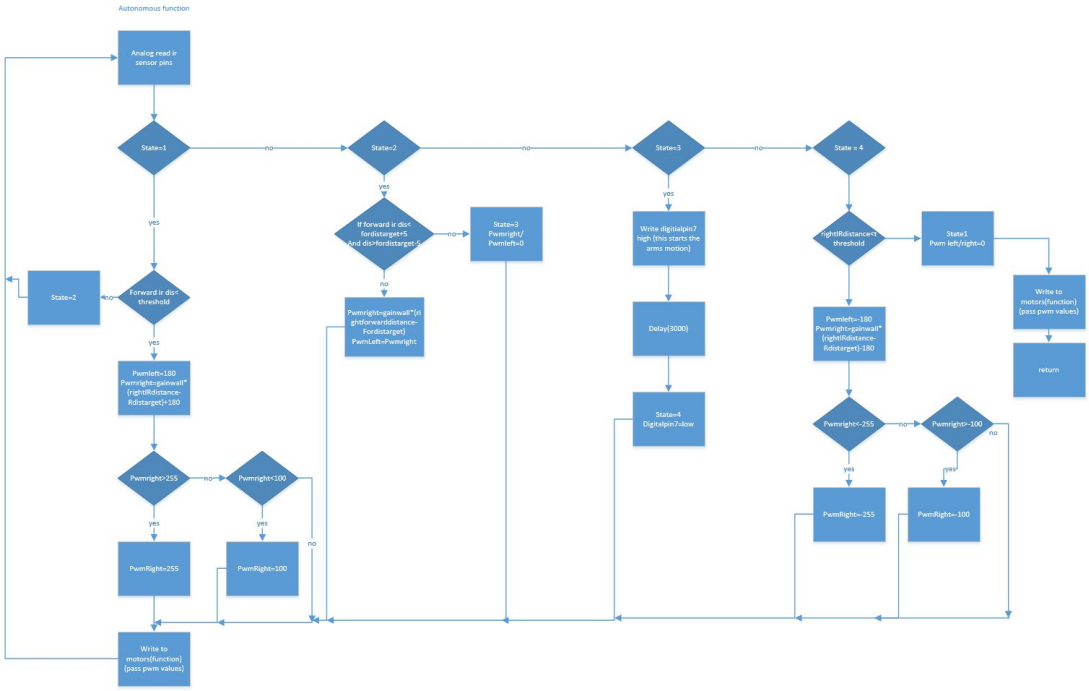


Figure 18: Autonomous Function

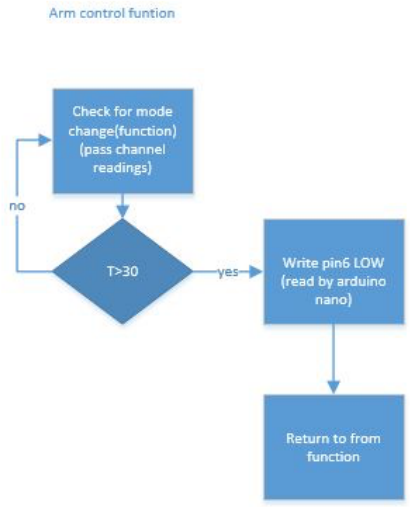


Figure 19: Arm Control Function

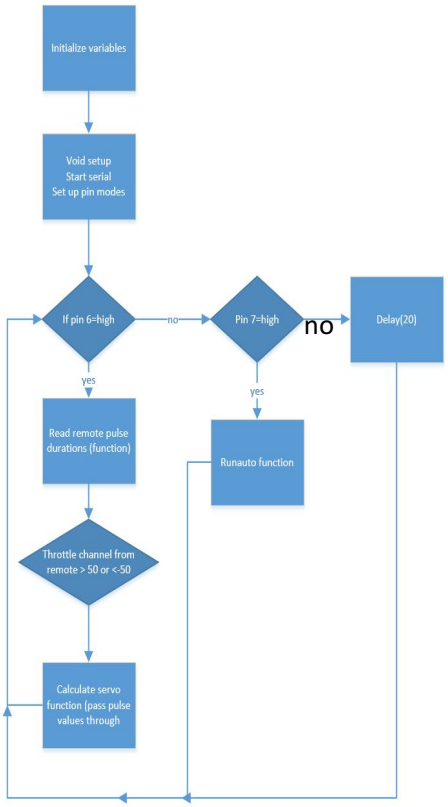


Figure 20: Second Arduino Main Loop

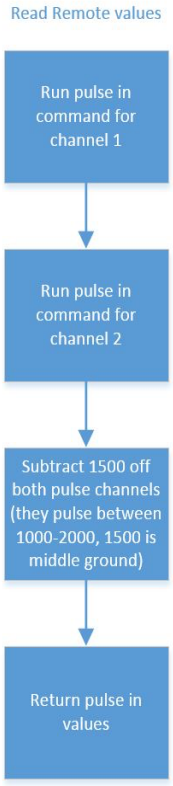


Figure 21: Second Arduino Read Remote Function

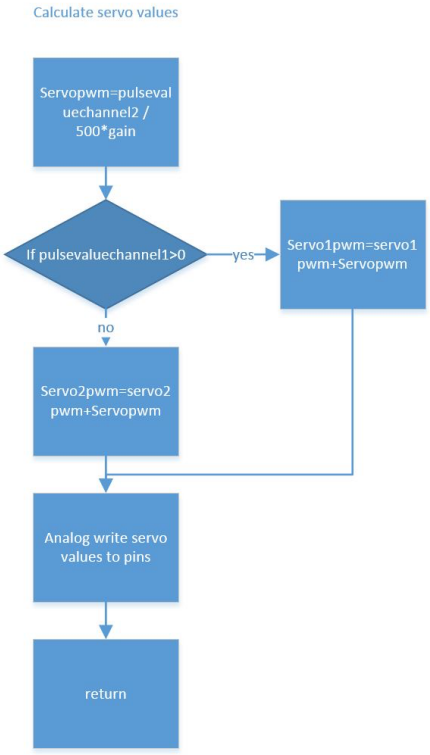


Figure 22: Second Arduino Calculate Servo Values Function

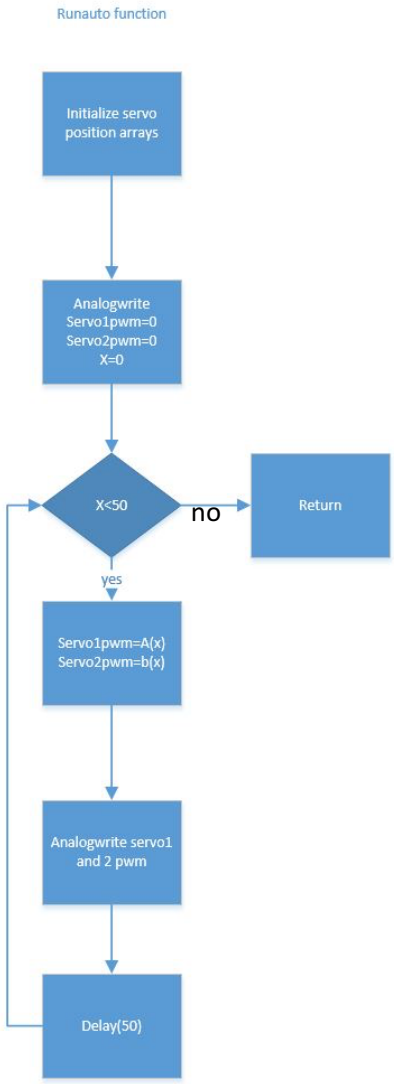


Figure 23: Second Arduino Runauto Function: Controls Arm During Autonomous Running

Arduino Uno Code

```
#define thrin 4
#define steerin 5
#define remotepin 6
#define autopin 8
#define  fwddis A2  //Analog distance sensors
#define  rhtdis A1  // Analog distance sensor
#define  motorRightPWM       3
#define  motorRightDirection 12
#define  motorLeftPWM       11
#define  motorLeftDirection 13
int steer=0;
int thr = 0;
int timer=0;
int Rightdis=0;
int Forwarddis=0;
int rightlim=150;
int forwardlim=150;
int forwardtgt=300;
int An=0;
int pwm1=0;
int pwm2=0;
float test1=0;
int mode=1;
int Righttgt=343;
int  fwdtracker=0;
int revtimer=0;

float gainwall=30;
float gainwall3=10;
float gainwall2=.02;
boolean remotestatus =false;
boolean automode= false;
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(remotepin, OUTPUT);
  pinMode(autopin, OUTPUT);
  pinMode(motorRightPWM, OUTPUT);
  pinMode(motorRightDirection, OUTPUT);
  pinMode(motorLeftPWM, OUTPUT);
  pinMode(motorLeftDirection, OUTPUT);
  pinMode(steerin, INPUT);
  pinMode(thrin, INPUT);
  Serial.begin (9600);
```

```
  digitalWrite(remotepin, LOW);
  digitalWrite(autopin, LOW);
  pinMode(fwddis, INPUT);
  pinMode(rhtdis, INPUT);
}

// the loop function runs over and over again forever
void loop() {
 if(automode==true){
  Rightdis=analogRead(rhtdis);
  Forwarddis= analogRead(fwddis);
  //Serial.print(Rightdis);
  //Serial.println("Rightdis");
  Serial.print(Forwarddis);
  Serial.println("Forwarddis");
  if (mode==1){
    autofwd();
    //Serial.println("auto 1");

  }
  if (mode==2){
    fwdtracker=0;
    revtimer=0;
    pwm1=0;
    pwm2=0;
    writetomotors(pwm1, 1, pwm2, 1);
    Serial.println("auto 2");
    digitalWrite(autopin, HIGH);
    delay(26000);
    digitalWrite(autopin, LOW);
    mode=3;
  }
  if(mode==3){
    Serial.println("auto 3");
    autorev();
  }

  An=0;

  }
 else{

    Rightdis=analogRead(rhtdis);
    Serial.print(Rightdis);
    Serial.println("Rightdis");
    if (Rightdis< rightlim){
      An=0;
```

```
  }
  else if(Rightdis> rightlim){
    //Serial.println("Rightdis");
   An=An+1;
  }
  if(An>100){
   automode=true;
   digitalWrite(remotepin, LOW);
  }
  readremote();
if(remotestatus==false){//driving car
      //Serial.println("driving car");
     drivecarclc();
     if (thr<100 && thr>-100){
      writetomotors(0, 1, 0, 1);
      //Serial.println("stop");
     }
     else if(thr>100){
       writetomotors(pwm1, 1, pwm2, 1);
       //Serial.println("fwd");
      }
      else if(thr<-100){
       writetomotors(pwm1, 0, pwm2, 0);
       //Serial.println("rev");
      }

}
if(remotestatus==true){//nano is driving arm
  //Serial.println("driving car");
 writetomotors(0, 0, 0, 0);
}
if(thr<100 && thr>-100){
  if(steer>100){

    timer=timer+1;
    Serial.println(timer);
    delay(100);
    if (timer==30){
      Serial.println("its time??????????????????");
      if (remotestatus==false){
           remotestatus=true;
           digitalWrite(remotepin, HIGH);

         }
         else{
           remotestatus=false;
           digitalWrite(remotepin, LOW);
```

```
        }
    }
    }
    else{
    timer=0;

   }

  }
  else{
      timer=0;
    }
  }
  delay(5);
}




////////////////////////////////////////////////////////////////////
void readremote() {
  steer = pulseIn(steerin, HIGH)-1470;
  thr = pulseIn(thrin, HIGH)-1470;
  //Serial.print(steer);
  //Serial.println("steer");
  //Serial.print(thr);
  //Serial.println("thr");
}

void autorev(){
    //Serial.println(1);
    if(rightlim<Rightdis){
    pwm1=-180+((Rightdis-Righttgt)*gainwall3);
    if(pwm1<-255){
            pwm1=-255;
          //  Serial.print("yes");
          }
          if(pwm1>-100){
            pwm1=-100;
          }
    pwm2=-180;
    writetomotors(pwm1, 1, pwm2, 1);
    }
    else{
      revtimer=revtimer+1;
      if (revtimer==40){
```

```
      automode=false;
      mode=1;
      pwm1=0;
      pwm2=pwm1;
      writetomotors(pwm1, 1, pwm2, 1);
    }
   }


 }


void autofwd(){
 if(forwardlim>Forwarddis){
  //Serial.println(1);
  if(rightlim<Rightdis){
  pwm1=180-((Rightdis-Righttgt)*gainwall);
  if(pwm1>255){
        pwm1=255;
       //  Serial.print("yes");
       }
       if(pwm1<100){
        pwm1=100;
       }
  pwm2=180;
  writetomotors(pwm1, 1, pwm2, 1);
  }


 }
 else{

  double distance=(forwardtgt-Forwarddis);
  //Serial.print(distance);
  //Serial.println("distance");

  double dist2=distance*gainwall2;
  //Serial.print(dist2);
  //Serial.println("dist2");
  double pwm1111=dist2*180;
  //Serial.print(pwm1111);
  //Serial.println("pwm1111");


  pwm1=pwm1111;
  if (Forwarddis>forwardtgt-15 && Forwarddis<forwardtgt+15){
   fwdtracker=fwdtracker+1;
```

```
    Serial.println(fwdtracker);
    if (fwdtracker==50){
      mode=2;
      pwm1111=0;
      writetomotors(pwm1, 1, pwm2, 1);
    }

  }
  else{
    fwdtracker=0;

  }
  if(pwm1111>180){
         pwm1111=180;
        //  Serial.print("yes");
        }
        if(pwm1111<-180){
          pwm1111=-180;
        }
  pwm1=pwm1111;
  pwm2=pwm1;
  Serial.print(pwm1);
  Serial.println(" pwm1");
  writetomotors(pwm1, 1, pwm2, 1);


 }

}



void drivecarclc() {
 float test1=0;
 float test2=0;
 float turn1=0;
 float turn2=0;
   test1= (thr/1.2);
   turn1= (steer/.5);
   pwm1=test1+turn1;
   pwm2=test1-turn1;
  // Serial.print(thr);
   //Serial.println("thr2");
   //Serial.print(steer);
   //Serial.println("steer2");
   //Serial.print(test1);
   //Serial.println("test1");
```

```
  //Serial.print(turn1);
  //Serial.println("turn1");
  //Serial.print(pwm1);
  //Serial.println("pwm1");

  //test2= (255*thr/500)
  //(steer/1.5);
  //pwm2=test2;
  //Serial.print(pwm2);
  //Serial.println("pwm2");
  if (pwm1>255){
    pwm1=255;
  }
  if (pwm1<-255){
    pwm1=-255;
  }
  if (pwm2>255){
    pwm2=255;
  }
  if (pwm2<-255){
    pwm2=-255;
  }


}


void writetomotors(int pwm11,int d1, int pwm22, int d2){
 int pwm12, pwm21;
 if (pwm11< 0){
   digitalWrite(motorLeftDirection, LOW);
 }
 else{
   digitalWrite(motorLeftDirection, HIGH);
 }
 pwm12=abs(pwm11);
 analogWrite(motorLeftPWM, pwm12);
 if (pwm22<0){
   digitalWrite(motorRightDirection, LOW);
 }
 else{
   digitalWrite(motorRightDirection, HIGH);
 }
 pwm21=abs(pwm22);
 analogWrite(motorRightPWM, pwm21);
 //Serial.print(pwm12);
   //Serial.println("pwm1atwheel");
```

```
    //Serial.print(pwm21);
    //Serial.println("pwm2atwheel");

}
```

Arduino Nano Code

```
#include <Servo.h>
Servo myservo1;
Servo myservo2;
Servo myservo3;

#define autopin 6
#define remotepin 5
#define steerin 4
#define thrin 3
#define mysin 2
int steer=0;
int thr=0;
int xx=0;
int servo1=50;
int servo2=50;
int servo3=50;
int servo2lim=120;
int gain1=.01;
int gain2=.01;
int gain3=.01;
int x=0;
int moves2[][3]={{50,120,50},
                 {10,120,10},
                 {10,10,10},
                 {50,10,50},
                 {50,10,50},
                 {50,10,50},
                 {50,10,50},
                 {50,10,50},
                 {50,10,50},
                 {30,30,30},
                 {50,10,50},
                 {50,10,50},
                 {50,10,50},
                 {50,10,50},
                 {50,10,50},
                 {50,10,50},
                 {50,10,50},
                 {50,10,50},
                 {50,10,50},
                 {50,10,50}};

void setup() {
  // put your setup code here, to run once:
  pinMode(steerin, INPUT);
```

```
  pinMode(thrin, INPUT);
  pinMode(mysin, INPUT);
  pinMode(remotepin, INPUT);
  pinMode(autopin, INPUT);
  //digitalWrite(remotepin, HIGH);
  //digitalWrite(autopin, HIGH);
  Serial.begin (9600);
  myservo1.attach(9);
  myservo2.attach(11);
  myservo3.attach(10);
  myservo1.write(servo1);
  myservo2.write(servo2);
  myservo3.write(servo3);


}

void loop() {
  // put your main code here, to run repeatedly:
  Serial.println(digitalRead(remotepin));
  if(digitalRead(remotepin)==1){
  readremote();
  servocalcremote();
  writeservo(servo1,servo2,servo3);
  x=0;
  delay(5);
  }
  else if(digitalRead(autopin)==1){
    if(x<5){
     servo1=(moves2[x][0]-servo1)/abs(moves2[x][0]-servo1)+servo1;
     servo2=(moves2[x][1]-servo2)/abs(moves2[x][1]-servo2)+servo2;
     servo3=(moves2[x][2]-servo3)/abs(moves2[x][2]-servo3)+servo3;
     writeservo(servo1,servo2,servo3);
     if(servo1==moves2[x][0] && servo2==moves2[x][1] && servo3==moves2[x][2]){
        x=x+1;
      }
     }
     delay(75);

  }
  else{
    x=0;
  }

}

void readremote() {
```

```
  steer = pulseIn(steerin, HIGH)-1470;
  thr = pulseIn(thrin, HIGH)-1470;


  }

void servocalcremote(){
  Serial.print(thr);
    Serial.println("thr");
     Serial.print(steer);
    Serial.println("steer");
  if(thr>50){
    if(steer>100){
      servo1=servo1+1;
      servo3=servo3+1;
    }
    if(steer<-100){
      servo1=servo1-1;
      servo3=servo3-1;
    }

    Serial.print(servo1);
    Serial.println("servo1");
    if(servo3<0){
      servo3=0;
    }
    if(servo3>175){
      servo3=175;
    }
    if(servo1<0){
      servo1=0;
      Serial.println("Whoops");
    }
    if(servo1>175){
      servo1=175;
    }
     Serial.print(servo1);
    Serial.println("servo1");
  }
  if(thr<-50){
    if(steer>100){
      servo2=servo2+1;
    }
    if(steer<-100){
      servo2=servo2-1;
    }
    if(servo2<0){
```

```
    servo2=0;
   }
   if(servo2>servo2lim){
    servo2=servo2lim;
   }
   Serial.print(servo2);
   Serial.println("servo2");
  }
}

void writeservo(int x,int y , int z){
  myservo1.write(x);
  if(y>servo2lim){
   y=servo2lim;
  }
  myservo2.write(y);
  myservo3.write(z);

  }
```