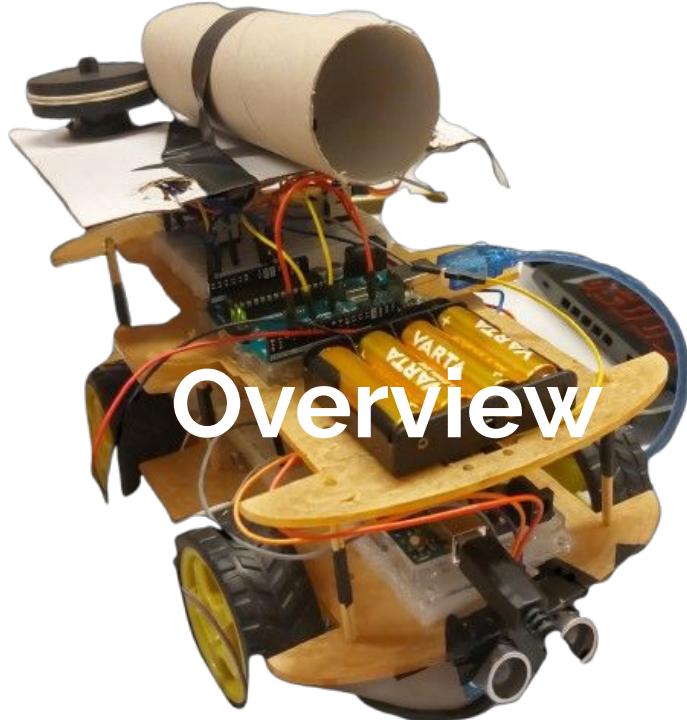

SIGNALS & MEASUREMENTS : Miniature Canon

IRIS KOSOVA
AZAT SABITOV
ALEX MONTES DE OCA



The project is a miniature robot whose main purpose is to detect an object within a certain rage, adjust the angle of the canon, and shoot a ping pong ball at the target.

Due to the large number of inputs needing connection to the Arduino, we decided to divide the work between two microcontrollers, choosing a master slave architecture

The architecture

Master

The master Arduino controls the basics of the robot, namely:

- The wheel motors
- The motor driver
- The distance sensor

Found on the lower platform for easy access to the driver and sensor

Slave

The slave Arduino controls the upper platform, meaning:

- The servo motor for tilting
- The motors of the canon
- The calculation of the angle

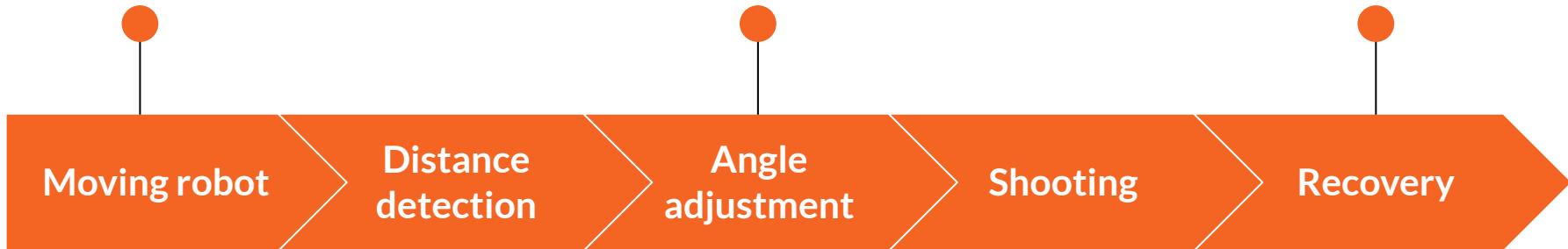
Dependent on the master for the shoot command

THE CODE

Based on the command given through the serial monitor, the direction of movement is chosen

The servo motor tilts the canon to the appropriate angle based on the distance of the target

After a 5-second delay, the wheels stop spinning and the robot can restart the process



The distance sensor detects the closest target within range

Once the angle is adjusted, the flywheels spin and the ball is fed manually

Moving the Robot

There are 9 commands that can be given through the Serial Monitor in order to control the movement of the robot, and communication between master and slave.

Connection with the slave

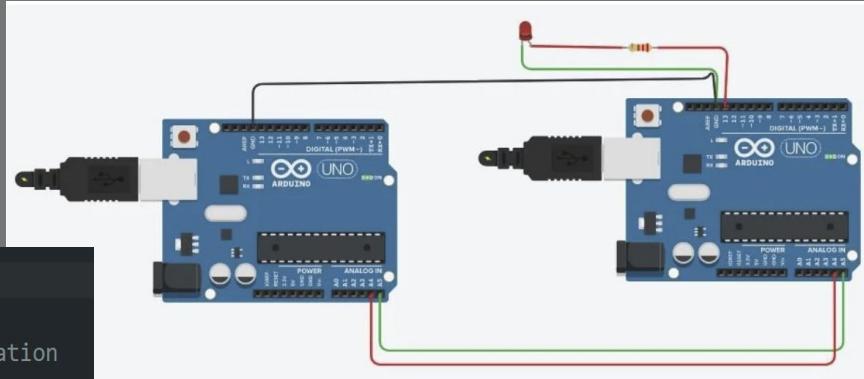
```
#include "Wire.h"
```

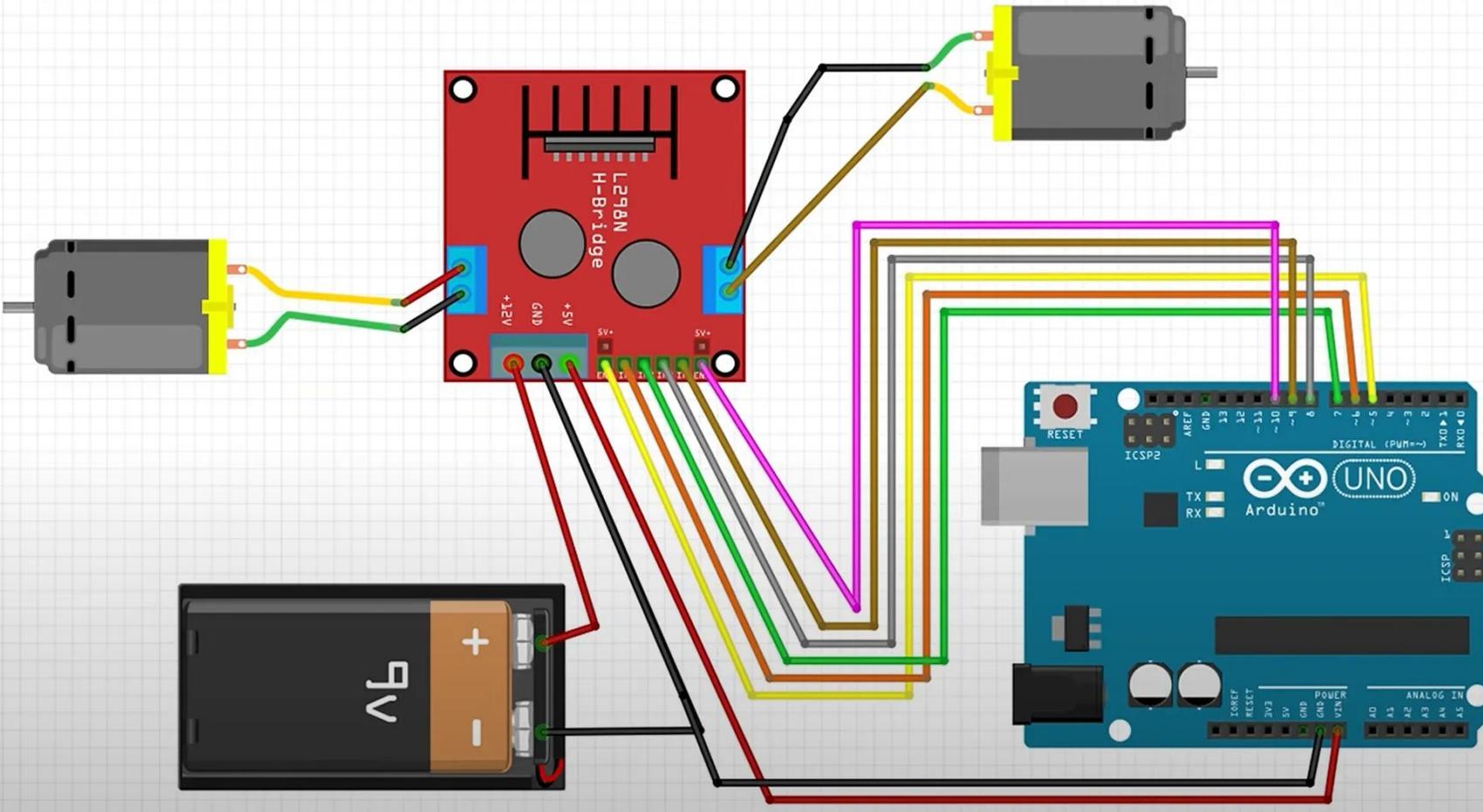
Using the wire library and connecting the A4, A5, and GND of both Arduinos for the communication setup.

```
const int slave_port = 2;
```

```
Wire.beginTransmission(slave_port);
Wire.write(lowByte(dist));
// transmission event, send distance for angle calculation
Wire.write(highByte(dist));
Wire.endTransmission();
```

The master can reach the slave using the wire library and the address of the slave.





FORWARD

```
if (inputvalue == 'F') { // forward
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
    digitalWrite(ena, 255);
    digitalWrite(in3, HIGH);
    digitalWrite(in4, LOW);
    digitalWrite(enb, 255);
}
```

BACKWARDS

```
else if (inputvalue == 'B') {
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
    digitalWrite(ena, 255);
    digitalWrite(in3, LOW);
    digitalWrite(in4, HIGH);
    digitalWrite(enb, 255);
}
```

TURN LEFT

```
else if (inputvalue == 'L') { //LEFT
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
    digitalWrite(ena, 255);
    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);
    digitalWrite(enb, 0);
}
```

TURN RIGHT

```
else if (inputvalue == 'R') {
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
    digitalWrite(ena, 0);
    digitalWrite(in3, HIGH);
    digitalWrite(in4, LOW);
    digitalWrite(enb, 255);
}
```

ROTATE CLOCKWISE

```
else if (inputvalue == 'A') {  
    digitalWrite(in1, HIGH);  
    digitalWrite(in2, LOW);  
    digitalWrite(ena, 255);  
    digitalWrite(in3, LOW);  
    digitalWrite(in4, HIGH);  
    digitalWrite(enb, 255);  
}
```

ROTATE COUNTERCLOCKWISE

```
else if (inputvalue == 'C') {  
    digitalWrite(in1, LOW);  
    digitalWrite(in2, HIGH);  
    digitalWrite(ena, 255);  
    digitalWrite(in3, HIGH);  
    digitalWrite(in4, LOW);  
    digitalWrite(enb, 255);  
}
```

STOP

```
else if (inputvalue == 'N') {  
    digitalWrite(in1, LOW);  
    digitalWrite(in2, LOW);  
    digitalWrite(ena, 0);  
    digitalWrite(in3, LOW);  
    digitalWrite(in4, LOW);  
    digitalWrite(enb, 0);
```

SEND

```
    else if(inputvalue == 'S') {  
        int16_t dist = distance;  
        // within range, the value can be casted to an integer (orig long)  
        Serial.println("output distance: ");  
        Serial.println(dist);  
        Wire.beginTransmission(slave_port);  
        Wire.write(lowByte(dist));  
        // transmission event, send distance for angle calculation  
        Wire.write(highByte(dist));  
        Wire.endTransmission();  
    }
```

Distance Detection

Ultrasonic Sensor in front of the car measures the nearest object in the range of [1; 200] cm. It was adjusted for feasible distance for our cannon based on practice

```
duration = pulseIn(echo, HIGH, 30000UL);
if(duration > 0) {
    distance = (duration * 0.03395) / 2;
    if (distance < maximumRange && distance > minimumRange) {
        Serial.println(distance);
    }
    else {
        Serial.println("Object out of range");
    }
}
```

Angle Adjustment

Based on the distance received from Master arduino, Slave arduino puts all the values in our derived formula and returns an angle that servo motor automatically sets

$$\theta = \frac{180}{\pi} \arctan \left(\frac{v^2 + \sqrt{|v^4 - g^2 d^2|}}{gd} \right)$$

Angle Calculation

```
// constants for the calculation
int init_vel = 1; // meters/s of the flywheel + motor thing
float g = 9.82;

int16_t calculateAngle(int16_t distance)
{
    // from the sensor's position, calculate angle needed to hit target
    float distMeters = (float)distance/100;

    // angle calculation, broken in numerator and denominator
    float num = pow(init_vel, 2) + sqrt(abs(pow(init_vel, 4) - sq(g) * sq(distMeters)));

    float den = g*distMeters;

    float ret = (atan(num/den)) * 180 / PI; // atan returns in rad, convert to deg

    Serial.println((String) "Unrounded calculated angle: " + ret);

    return round(ret); // servo.write only takes int
}
```

Constants and Invoked functions/libraries

```
#include "Servo.h"
#include "Wire.h"
#include "Math.h"

Servo servo;

int servoPin = 8;
int speed = 225;

void prepareCannon(int16_t angle)
{
    for(int x = 0; x <= angle; x++)
    {
        servo.write(x);
        delay(10);
    }
    Serial.println((String)"58 prepare cannon arg angle: "+ angle);
```

SHOOTING

```
pinMode(motorPin1, OUTPUT);
pinMode(motorPin2, OUTPUT);

int motorPin1 = 3;
int motorPin2 = 6;
```

First, we declare and set up the pins of the motors to be used.

```
int motorState = 0; // 0: off, 1: on
unsigned long motorStartTime = 0;
const long motorDuration = 5000; // 5 seconds to shoot
int speed = 200;
```

Various variables control the shooting process:

- State of the motor (on/off)
- Start time (redefined later with millis())
- Duration of the shooting
- Shooting speed (0-255)

When an instruction is received from the master ('S'), we prepare to shoot by adjusting the angle, turning the motors on to the speed set beforehand, and updating the motor start time and motor state.

```
void receiveEvent(int howMany) {
    if(howMany >= 2) {
        uint8_t lsb = Wire.read();
        uint8_t msb = Wire.read();
        received = (int16_t)( (msb << 8) | lsb ); // reconstruct LSB-first

        Serial.println((String) "distance received from master: " + received);

        int16_t angle = calculateAngle(received);
        Serial.println((String)"Calculated Angle: " + angle);

        prepareCannon(angle);
        motorOn(speed); // arg is speed
        motorStartTime = millis(); // record current time
        motorState = 1; // check loop()

        //Serial.println("-----"); // for cleaner serial output
    }
    else if(howMany == 1) {
        Wire.read(); // handles single byte msgs or error
    }
}
```

```
void motorOn(int speed)
{
    analogWrite(motorPin1, speed);
    analogWrite(motorPin2, speed); // turn motors on
    Serial.println("76: Motors turned on.");
}

void motorOff()
{
    digitalWrite(motorPin1, LOW);
    digitalWrite(motorPin2, LOW); // turn motors off
}
```

Functions turn motors on/off simultaneously.

```
void loop() {
    // Check how long motor has been running
    if(motorState == 1) {
        if(millis() - motorStartTime >= motorDuration) {
            motorOff();
            motorState = 0;
            Serial.println("105: Motors turned off.");
            Serial.println("-----");
        }
    }

    delay(400);
}
```

loop() checks how long the motors have been via the motor state. After the defined time, the motors are turned off and the process is restarted.