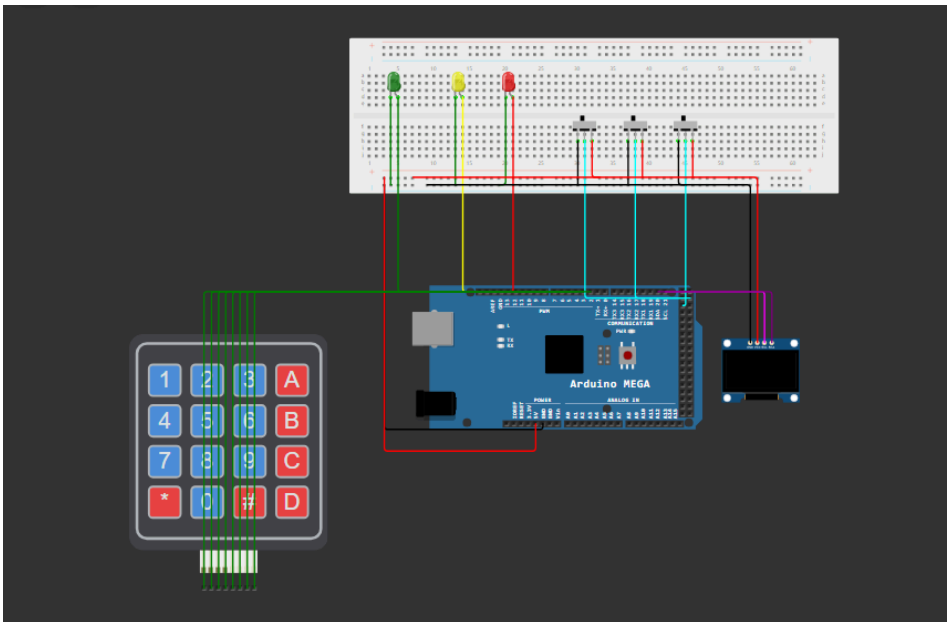
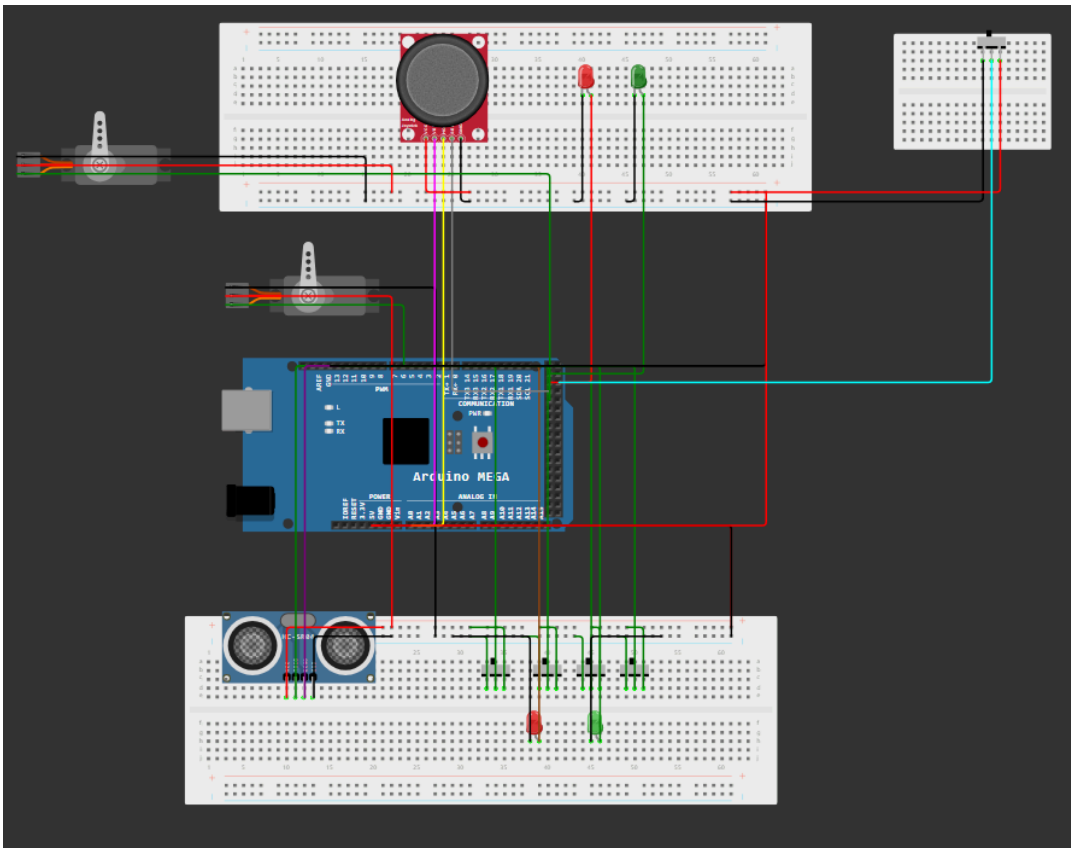
Schaltplan - Projektwoche 2024

Der Code und Schaltplan des **ersten Arduino** findet man hier. Wichtig zu beachten ist, dass die Schalter, die Bananenstecker repräsentieren. Zudem werden hier nicht die Steckverbindungen für die Signalübertragung an **Arduino2** und **Arduino3** gezeigt. Diese sind jedoch einfach mit Kabeln verbunden (siehe im Code für die Steckplätze. Achtung: Unterschiedliche bei den jeweiligen Arduino, also nicht selbe Pinbelegung bei zwei Arduino nehmen!)



Der Code und Schaltplan des **zweiten Arduino** findet man hier. Wichtig zu beachten ist, dass der Schalter, das Signal, also die Verbindung zu **Arduino1** darstellt. Zudem hat der Joystick einen anderen Code, da das verbaute Gerät andere x- und y-Werte hat (andere Schwellenwerte), als in der Simulation. Zusätzlich ist **SignalPin = 23**, und ***nicht*** **SignalPin = 25**. Die Reihenfolge beim echten Arduino ist zudem: Oben, Rechts, Unten und Links. In der Simulation ist es Oben, Rechts, Links und Unten. Der ***Code*** für den **zweiten Arduino** findet man auf der nächsten Seite.



Das ist der ***Code*** für **Arduino2**:

```
#include <Servo.h>

// Pin definitions for Code1
const int trigPin = 9;
const int echoPin = 10;
const int greenLedPin1 = 13; // Grüne LED für Code1
const int redLedPin1 = 12; // Rote LED für Code1
const int switchPins[] = {2, 3, 4, 5}; // Schalter-Pins für Code1
const int servoPin1 = 6; // Servomotor Pin für Code1

// Pin definitions for Code2
const int xPin = A0; // Analog pin for X-axis (Joystick X)
const int yPin = A1; // Analog pin for Y-axis (Joystick Y)
const int swPin = 26; // Digital pin for Joystick button (SW pin)
const int greenLedPin2 = 22; // Green LED für Code2
```

```
const int redLedPin2 = 24; // Red LED für Code2
const int servoPin2 = 28; // Servomotor Pin für Code2

// Pin definition for Signal reception
const int signalPin = 23; // Pin, an dem das Signal vom ersten Arduino empfangen wird


Servo myservo1;
Servo myservo2;

// Bewegungsabfrage für Code2
int stepCount = 0;  // Schrittzähler (0 = kein Schritt)
bool greenLedOn = false;  // Flag, ob die grüne LED bereits an ist

// Joystick Schwellenwerte
int lowerThresholdX = 550;
int upperThresholdX = 650;
int lowerThresholdY = 250;
int upperThresholdY = 350;

// Aktueller Schritt in der Reihenfolge
enum Step { STEP_UP, STEP_RIGHT, STEP_DOWN, STEP_LEFT };
Step currentStep = STEP_UP;  // Starten mit dem ersten Schritt (Oben)

void setup() {
  // Setup für Code1
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(greenLedPin1, OUTPUT);
  pinMode(redLedPin1, OUTPUT);
```

```
  for (int i = 0; i < 4; i++) {
    pinMode(switchPins[i], INPUT_PULLUP);
  }
  myservo1.attach(servoPin1);
  myservo1.write(0); // Sicherstellen, dass der Servo in der Position 0 ist

  //Checken, dass alle LED's aus sind
  digitalWrite(redLedPin1, LOW);
  digitalWrite(redLedPin2, LOW);
  digitalWrite(greenLedPin1, LOW);
  digitalWrite(greenLedPin1, LOW);

  // Setup für Code2
  pinMode(greenLedPin2, OUTPUT);
  pinMode(redLedPin2, OUTPUT);
  pinMode(swPin, INPUT_PULLUP);
  myservo2.attach(servoPin2);
  myservo2.write(0); // Sicherstellen, dass der Servo in der Position 0 ist

  // Setup für Signalempfang
  pinMode(signalPin, INPUT);

  // Seriellen Monitor starten
  Serial.begin(9600);
}

void loop() {
  if (digitalRead(signalPin) == HIGH) { // Prüfen, ob das Signal an signalPin HIGH ist
    Serial.println("Signal empfangen");
```

```
    runCode1();
  } else {
    // Debug-Ausgabe, wenn kein Signal empfangen wird
    Serial.println("Warten auf Signal");
    delay(1000); // Warten, um die serielle Ausgabe nicht zu überfluten
  }
}

void runCode1() {
  // Code1 Logik (Abstand messen)
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH);
  long distance = duration * 0.34 / 2;

  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" mm");

  if (distance <= 170) {
    int switchStates[4];
    for (int i = 0; i < 4; i++) {
      switchStates[i] = digitalRead(switchPins[i]);
    }
```

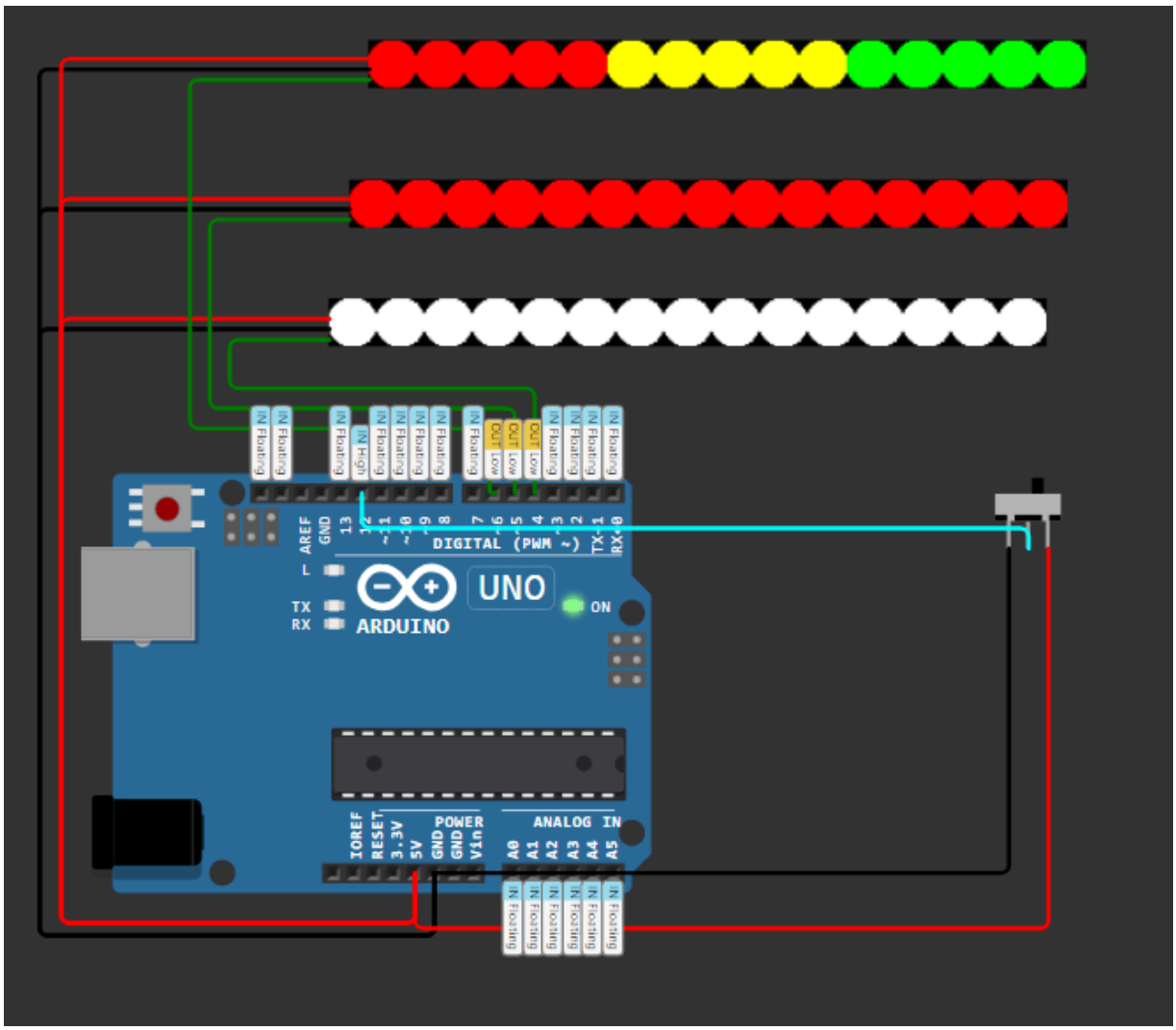
```
  if (switchStates[0] == HIGH && switchStates[1] == HIGH && switchStates[2] == LOW && switchStates[3] == HIGH) {
    digitalWrite(greenLedPin1, HIGH); // Grüne LED an für Code1
    digitalWrite(redLedPin1, LOW); // Rote LED aus für Code1
    myservo1.write(90); // Servo auf 90 Grad für Code1
    runCode2(); // Code2 aufrufen
  } else {
    digitalWrite(greenLedPin1, LOW); // Grüne LED aus für Code1
    digitalWrite(redLedPin1, HIGH); // Rote LED an für Code1
    myservo1.write(0); // Servo auf 0 Grad für Code1
  }
 } else {
  digitalWrite(greenLedPin1, LOW); // Grüne LED aus, wenn Abstand größer als 100 mm
  digitalWrite(redLedPin1, LOW); // Rote LED aus, wenn Abstand größer als 100 mm
  myservo1.write(0); // Servo auf 0 Grad für Code1
 }

 delay(100);
}

void runCode2() {
 // Code2 Logik (Joystick Bewegungen)
 int xValue = analogRead(xPin);
 int yValue = analogRead(yPin);
 bool buttonPressed = digitalRead(swPin) == LOW;  // Joystick Button gedrückt?

 // Debug-Ausgabe der aktuellen Joystick-Werte
 Serial.print("Joystick X: ");
```

```
Serial.print(xValue);
Serial.print(" Joystick Y: ");
Serial.println(yValue);

if (greenLedOn) {
  // Wenn die grüne LED an ist, nichts mehr tun
  return;
}

// Überprüfen der aktuellen Joystick-Bewegung
if (currentStep == STEP_UP && yValue > upperThresholdY) {
  stepCount++;
  Serial.println("Step: Oben");
  digitalWrite(redLedPin2, LOW);  // Rote LED aus, falls vorher an
  delay(500);  // Kurze Pause nach dem Schritt
  currentStep = STEP_RIGHT;  // Nächster Schritt
} else if (currentStep == STEP_RIGHT && xValue < lowerThresholdX) {
  stepCount++;
  Serial.println("Step: Rechts");
  digitalWrite(redLedPin2, LOW);  // Rote LED aus, falls vorher an
  delay(500);
  currentStep = STEP_DOWN;  // Nächster Schritt
} else if (currentStep == STEP_DOWN && yValue < lowerThresholdY) {
  stepCount++;
  Serial.println("Step: Unten");
  digitalWrite(redLedPin2, LOW);  // Rote LED aus, falls vorher an
  delay(500);
  currentStep = STEP_LEFT;  // Nächster Schritt
```

```
  } else if (currentStep == STEP_LEFT && xValue > upperThresholdX) {
    stepCount++;
    Serial.println("Step: Links");
    digitalWrite(redLedPin2, LOW);  // Rote LED aus, falls vorher an
    delay(500);
    greenLedOn = true;
    digitalWrite(greenLedPin2, HIGH);  // Grüne LED an für Code2
    myservo2.write(90);  // Servo auf 90 Grad drehen für Code2
    Serial.println("Alle Schritte abgeschlossen!");
  } else if (buttonPressed) {
    // Reset: Wenn der Joystick-Button gedrückt wird, zurücksetzen
    currentStep = STEP_UP;
    stepCount = 0;
    greenLedOn = false;
    digitalWrite(greenLedPin2, LOW);  // Grüne LED ausschalten
    myservo2.write(0);  // Servo auf 0 Grad zurücksetzen
    Serial.println("Reset durchgeführt");
  }

  // Wenn die Schritte nicht in der richtigen Reihenfolge gemacht werden, rote LED an
  if (stepCount != 4 && !greenLedOn) {
    digitalWrite(redLedPin2, HIGH);  // Rote LED an
  } else {
    digitalWrite(redLedPin2, LOW);  // Rote LED aus, wenn die Reihenfolge korrekt ist
  }
}
```

Der Code und Schaltplan des **dritten Arduino** findet man hier. Bei der Simulation können nicht alle LED's angeschlossen werden, weshalb sie als verkürzte Form dargestellt werden. Der Schalter repräsentiert das

Signal von **Arduino1**. Außerdem ist die Zeit der Spieldauer (pro Minute) in der Simulation auf ***600ms*** gestellt, um sie besser darzustellen. Bei dem echten Arduino liegt sie bei ***60000ms***.



Das ist der komplette Schreibplan unserer Projektwoche bei ***Think-Square*** in Bochum 2024.

Projekt: **Rätselwürfel**

An dem Code waren beteiligt: **Tom, Gabriel und Liam**

Der Feinschliff, der Hauptverfasser des Codes und des Schreibplans: **Liam (*Grüße an ChatGPT*)**

Gebaut wurde der Würfel von: **Marvin, Julian, Iyad, Emir, Leonardo, ...**

Die Story haben geschrieben: **Tommy, Nico, Selina, Romaissa**

Alle sind/waren Schüler der Q2 des Jahrgangs 2024/2025 der Gesamtschule Erle

Fragen zu dem Schaltplan/Würfel an: liam.kollaczek@gmail.com