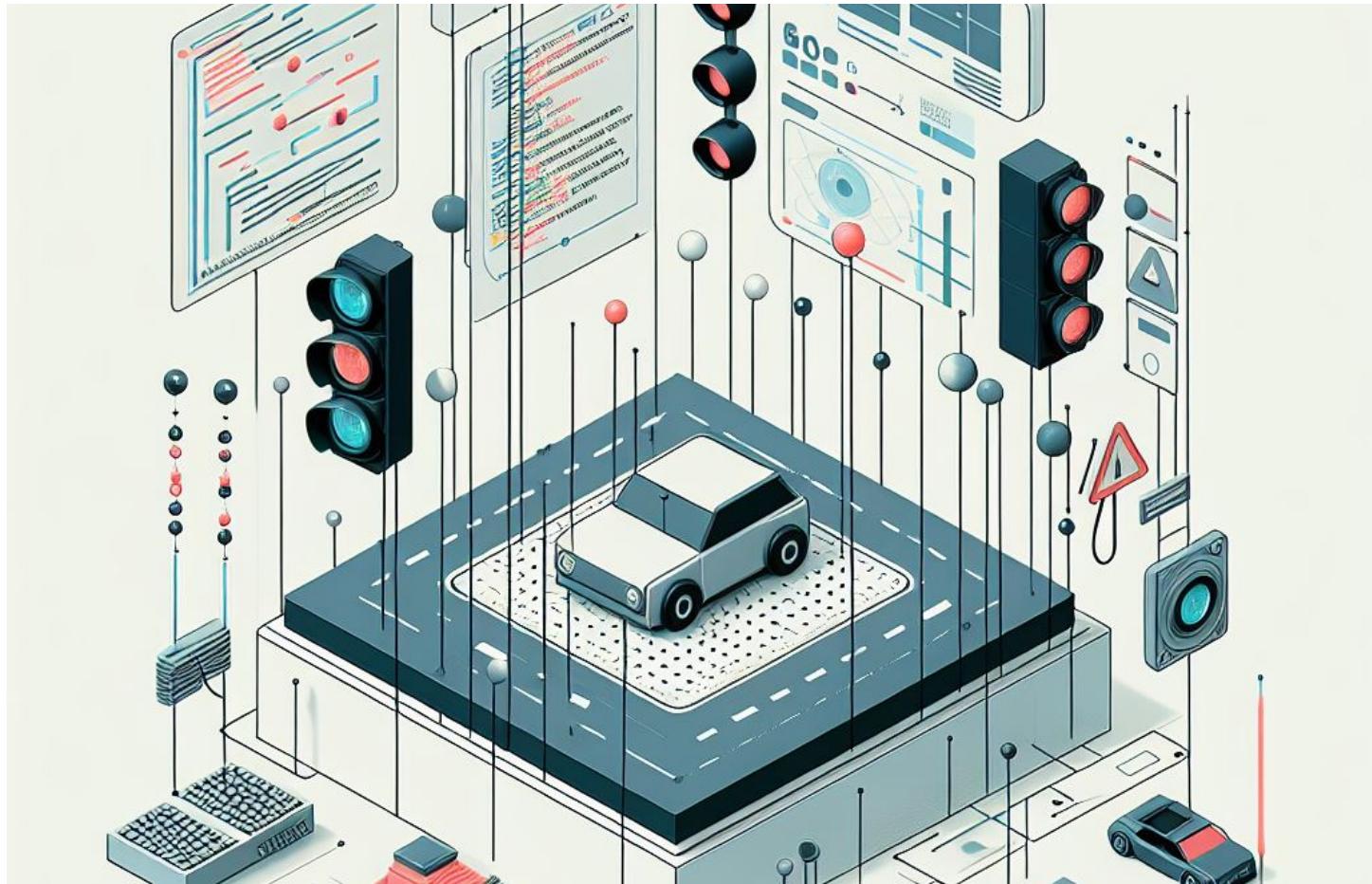


TRAFFIC LIGHT CONTROLLER



Projekt-Abgabe Arduino
Lernfeld 7: Cyberphysische Systeme
Abgabe: 19.03.2024

Realisierung einer Ampelanlage mittels Arduino-Hardware, bestehend aus einer Fahrzeugampel (rot, gelb, grün) und einer Fußgängerampel.

Diese Anlage erfordert spezifische Schaltungen und Reaktionen auf Tasterbetätigungen sowie einen Näherungssensor für Fahrzeugerkennung.

Auszubildender: Cristian Felipe Castillo Barrero

Lehrer : Peter Grüning

Ausbildungsbetrieb: Deutsche Börse AG

Vorwort

Die Implementierung erfolgte mittels eines endlichen Zustandsautomaten (Finite State Machine, FSM). Der Einsatz eines FSM wurde gewählt, da er eine klare Strukturierung der verschiedenen Zustände ermöglicht und eine einfache Handhabung von Zustandsübergänge bietet.

Vorteile eines FSM

- Klare Struktur
- Einfache Handhabung von Zustandsübergänge
- Modularität und Erweiterbarkeit
- Verständlichkeit und Nachvollziehbarkeit

Die Entscheidung, einen FSM für die Implementierung dieses Projektes zu verwenden, basiert auf der Überlegung, dass diese Methode die besten Voraussetzungen für eine übersichtliche, strukturierte und gut wartbare Lösung bietet.

Durch die klare Definition der Zustände wird eine robuste und zuverlässige Steuerung des Ampelsystems erreicht.

Aufgabenstellung

Ampelanlage

Eine Ampelanlage besteht aus einer Fahrzeugampel (rot, gelb, grün) und einer Fußgängerampel.

Anforderungen

- Beim Start der Anlage ist die Fahrzeugampel grün, die Fußgängerampel rot.
- Per Taster kann ein Fußgänger eine Grün-Phase für die Fußgänger und eine Rot-Phase für Fahrzeuge auslösen.
- Vor der Rot-Phase für Fahrzeuge hat die Fahrzeugampel eine Gelb-Phase.
- Die Gelb-Phase dauert zwei Sekunden.
- Eine Rot-Phase für Fahrzeuge dauert immer 15 Sekunden.
- Nach der Rot-Phase für Fahrzeuge erfolgt die Rot-Gelb-Phase (drei Sekunden) und danach wieder die Grün-Phase.
- Die Grün-Phase für Fußgänger beginnt sofort mit der Rot-Phase für Fahrzeuge.
- Die Rot-Phase für Fußgänger beginnt drei Sekunden vor der Rot-Gelb-Phase für Fahrzeuge.
- Die Gelb-Phase für Fahrzeuge beginnt zehn Sekunden nach Drücken des Fußgänger-Tasters, wenn die Grün-Phase der Fahrzeuge mindestens 30 Sekunden andauerte
- War die Grün-Phase für Fahrzeuge weniger als 30 Sekunden, beginnt die Gelb-Phase nach Drücken des Fußgänger-Tasters und wenn die Grün-Phase mindestens 30 Sekunden angedauert hatte.
- Ein Näherungssensor soll prüfen, ob sich Fahrzeuge über eine bestimmte Linie in der Fahrzeug-Rot- Phase bewegen. In dem Fall soll eine Lampe aufblitzen.

Programmcode

```
/*
 * TrafficLightController
 *
 * The program simulates a basic traffic light system with green, yellow, red, and red-yellow phases
 * for cars, along with corresponding pedestrian signals.
 *
 * Developed by: Cristian Felipe Castillo Barrero
 * Date: 2024-03-19
 *
 * Pin 10: Red LED for Cars
 * Pin 9: Yellow LED for Cars
 * Pin 8: Green LED for Cars
 * Pin 3: Red LED for Pedestrians
 * Pin 2: Green LED for Pedestrians
 * Pin 7: LED for Flashing
 * Pin 5: Push Button
 * Pin 12: Trigger Pin for Distance Sensor
 * Pin 13: Echo Pin for Distance Sensor
 * Pin GND (Ground): ground rail (-) on the breadboard
 * Pin 5V: (+) rail on the breadboard
 */
const int redCar = 10;      // Pin for the red light of cars
const int yellowCar = 9;    // Pin for the yellow light of cars
const int greenCar = 8;     // Pin for the green light of cars

const int redWalkers = 3;   // Pin for the red light of pedestrians
const int greenWalkers = 2; // Pin for the green light of pedestrians

const int flashLED = 7;    // Pin for the flash LED

const int pushButton = 5;  // Pin for the push button

const int dSensorTrig = 12; // Pin for the trigger of distance sensor
const int dSensorEcho = 13; // Pin for the echo of distance sensor

enum TrafficLightStates {GREEN_PHASE, YELLOW_PHASE, RED_PHASE, RED_YELLOW_PHASE};

// Initial state: Green phase for cars
TrafficLightStates trafficLightState = GREEN_PHASE;

int buttonState;

// Time stamp variables
unsigned long lastGreenPhaseTimeStamp = 0;
unsigned long milliCounter; // Required for delay
unsigned long greenPhaseDuration;
```

```

/** 
 * @brief Generate a trigger signal for the distance sensor and check if the distance is lower than
10cm to activate a blitzer.
*
* @param None.
*
* @details This function sets the trigger pin of the distance sensor to HIGH to initiate a
measurement signal,
* delays for a short duration to ensure the trigger signal is stable, and then sets the trigger pin
back to LOW
* to end the measurement signal. It then checks if the distance is lower than 10cm and activates a
blitzer accordingly.
*
* @note Ensure that the appropriate pins are configured for the distance sensor, blitzer LED, and
that the appropriate delay is applied.
*
* @return None.
*/
void triggerAndCheckDistance()
{
    // Set the trigger pin of the distance sensor to HIGH to initiate a measurement signal.
    digitalWrite(dSensorTrig, HIGH);
    // Delay for a short duration to ensure the trigger signal is stable.
    delayMicroseconds(1000);
    // Set the trigger pin of the distance sensor back to LOW to end the measurement signal.
    digitalWrite(dSensorTrig, LOW);

    // If the distance is lower than 10 cm, activate blitzer
    if (pulseIn(dSensorEcho, HIGH) / 58 <= 10)
    {
        digitalWrite(flashLED, HIGH);
        delay(10);
        digitalWrite(flashLED, LOW);

        // Wait 1 second for the next action
        delay(1000);
    }
}

/** 
 * @brief Set the traffic lights to display the green phase.
*
* @param None.
*
* @details This function turns on the green light for cars and the red light for pedestrians,
indicating the green phase.
*
* @note Ensure that the appropriate pins are configured for the traffic lights.
*
* @return None.
*/
void greenPhase()
{
    digitalWrite(redCar, LOW);
    digitalWrite(yellowCar, LOW);
    digitalWrite(greenCar, HIGH);

    digitalWrite(redWalkers, HIGH);
    digitalWrite(greenWalkers, LOW);
}

```

```

/**
 * @brief Set the traffic lights to display the yellow phase.
 *
 * @param None.
 *
 * @details This function turns on the yellow light for cars and the red light for pedestrians,
indicating the yellow phase.
* It then delays for 2 seconds before transitioning to the red phase.
*
* @note Ensure that the appropriate pins are configured for the traffic lights.
*
* @return None.
*/
void yellowPhase()
{
    digitalWrite(redCar, LOW);
    digitalWrite(yellowCar, HIGH);
    digitalWrite(greenCar, LOW);

    digitalWrite(redWalkers, HIGH);
    digitalWrite(greenWalkers, LOW);

    delay(2000);
    trafficLightState = RED_PHASE;
}

/**
* @brief Set the traffic lights to display the red phase.
*
* @param None.
*
* @details This function turns on the red light for cars and the green light for pedestrians,
indicating the red phase.
* It activates a blitzer for cars for 15 seconds.
* After that, it transitions to the red-yellow phase.
*
* @note Ensure that the appropriate pins are configured for the traffic lights and sensors.
*
* @return None.
*/
void redPhase()
{
    digitalWrite(redCar, HIGH);
    digitalWrite(yellowCar, LOW);
    digitalWrite(greenCar, LOW);

    digitalWrite(redWalkers, LOW);
    digitalWrite(greenWalkers, HIGH);

    // Speed camera //
    milliCounter = millis();
    while (millis() - milliCounter <= 12000)
    {
        triggerAndCheckDistance();
    }

    digitalWrite(redWalkers, HIGH);
    digitalWrite(greenWalkers, LOW);

    // Speed camera //
    milliCounter = millis();
    while (millis() - milliCounter <= 3000)
    {
        triggerAndCheckDistance();
    }

    trafficLightState = RED_YELLOW_PHASE;
}

```

```

/**
 * @brief Set the traffic lights to display the red-yellow phase.
 *
 * @param None.
 *
 * @details This function turns on both the red and yellow lights for cars and the red light for
pedestrians, indicating the red-yellow phase.
 * It then delays for 3 seconds before transitioning back to the green phase and updating the
timestamp.
 *
 * @note Ensure that the appropriate pins are configured for the traffic lights.
 *
 * @return None.
 */
void redYellowPhase()
{
    digitalWrite(redCar, HIGH);
    digitalWrite(yellowCar, HIGH);
    digitalWrite(greenCar, LOW);

    digitalWrite(redWalkers, HIGH);
    digitalWrite(greenWalkers, LOW);

    // Speed camera //
    milliCounter = millis();
    while (millis() - milliCounter <= 3000)
    {
        triggerAndCheckDistance();
    }

    trafficLightState = GREEN_PHASE;
    lastGreenPhaseTimeStamp = millis();
}

/**
 * @brief Set up the pin modes for the components used in the traffic light system.
 *
 * @param None.
 *
 * @details This function initializes the pin modes for the push button, car traffic lights (red,
yellow, green),
* pedestrian traffic lights (red, green), flash LED, and distance sensor (trigger and echo pins).
* It configures the push button pin as INPUT_PULLUP to enable internal pull-up resistor.
* Other pins are set as OUTPUT for controlling various components.
*
* @note Make sure to connect the components to the correct pins specified in the global variables.
*
* @return None.
*/
void setup()
{
    pinMode(pushButton, INPUT_PULLUP);

    pinMode(redCar, OUTPUT);
    pinMode(yellowCar, OUTPUT);
    pinMode(greenCar, OUTPUT);

    pinMode(redWalkers, OUTPUT);
    pinMode(greenWalkers, OUTPUT);

    pinMode(flashLED, OUTPUT);

    pinMode(dSensorTrig, OUTPUT);
    pinMode(dSensorEcho, INPUT);
}

```

```

/**
 * @brief Main loop function to control the traffic light system.
 *
 * @param None.
 *
 * @details This function continuously loops through different traffic light phases based on the current state.
 * It calls specific functions for each phase (green, yellow, red, red-yellow).
 * It also checks for button press to manually trigger the transition to the yellow phase.
 * If the green phase duration exceeds or reaches 30 seconds, it waits for 10 seconds before transitioning to yellow.
 * If the button is pressed during the green phase (less than 30 seconds), the time difference is calculated and used to determine the delay before switching to yellow.
 *
 * @note Ensure that the button pin is correctly connected and configured.
 *
 * @return None.
 */
void loop()
{
    switch(trafficLightState)
    {
        case GREEN_PHASE:
            greenPhase();
            break;
        case YELLOW_PHASE:
            yellowPhase();
            break;
        case RED_PHASE:
            redPhase();
            break;
        case RED_YELLOW_PHASE:
            redYellowPhase();
            break;
    }

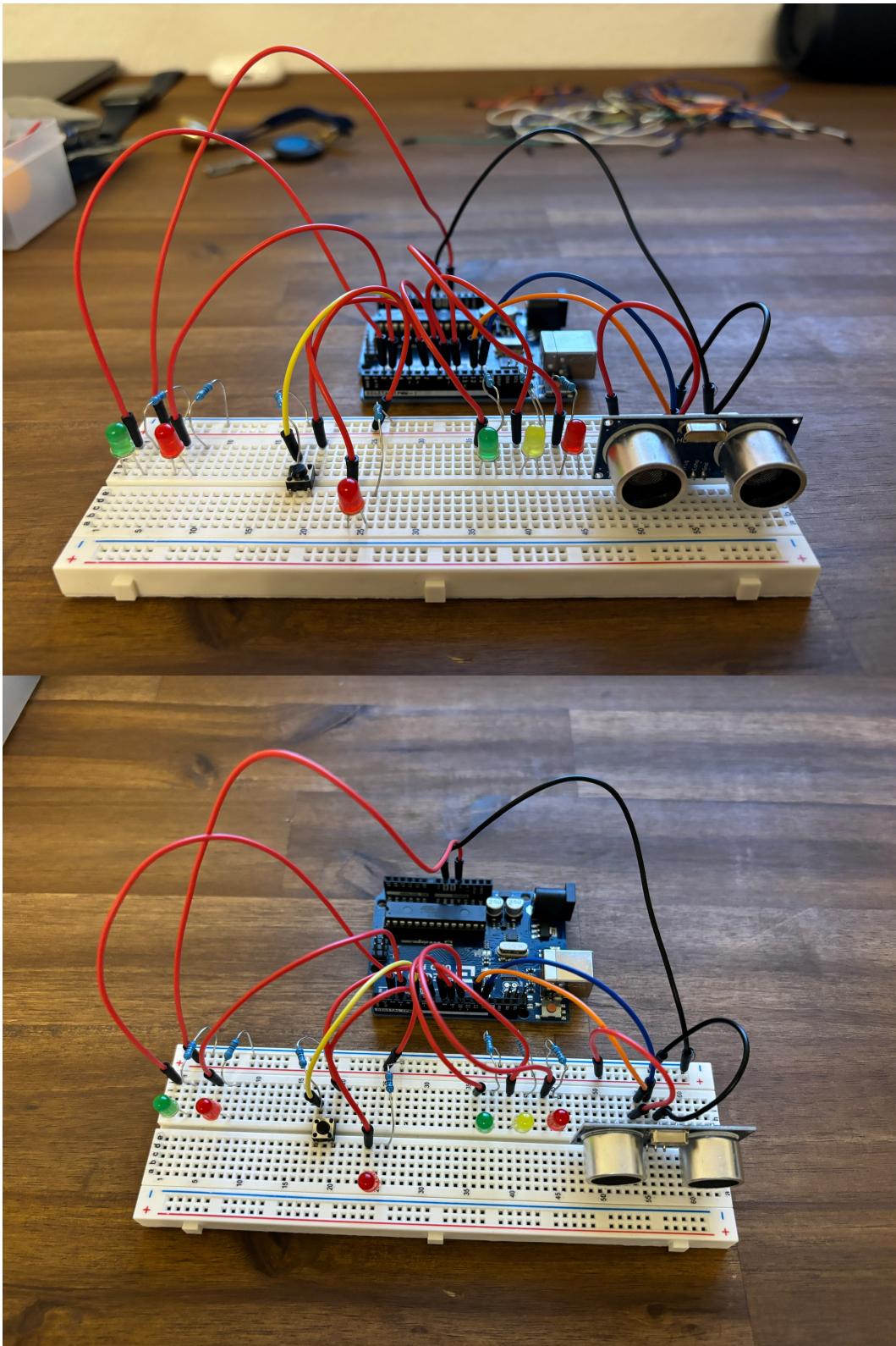
    // Check button state to manually trigger transition to yellow phase
    buttonState = digitalRead(pushButton);

    // Transition to yellow phase based on button press and green phase duration
    if (buttonState == HIGH)
    {
        // Calculate the duration of the current green phase
        unsigned long greenPhaseDuration = millis() - lastGreenPhaseTimeStamp;

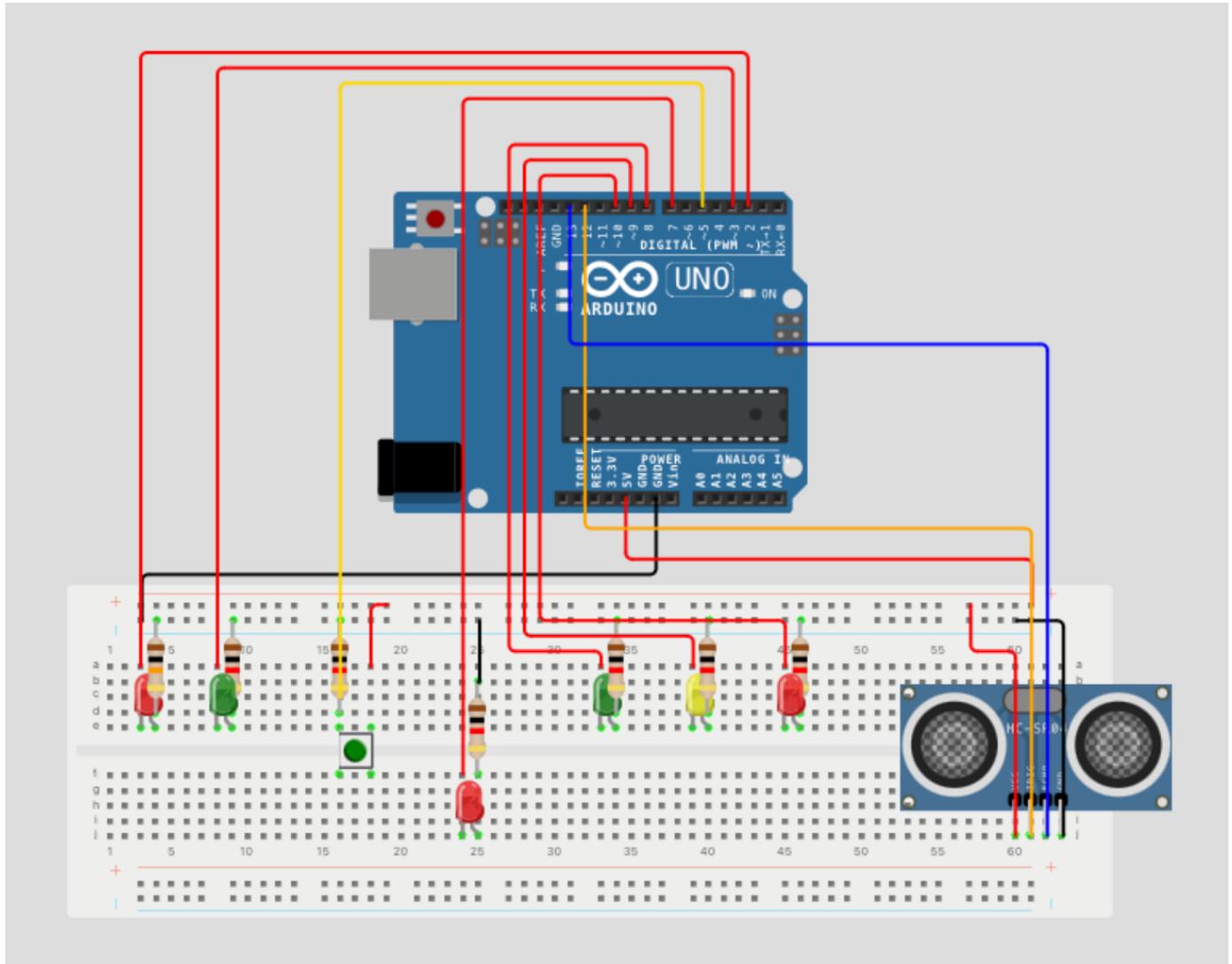
        // If green phase duration exceeds or reaches 30 seconds, wait 10 seconds before transitioning to yellow
        if (greenPhaseDuration >= 30000)
        {
            delay(10000);
            trafficLightState = YELLOW_PHASE;
        }
        // If button is pressed before green phase duration reaches 30 seconds, calculate remaining time and wait before transitioning to yellow
        else
        {
            delay(30000 - greenPhaseDuration);
            trafficLightState = YELLOW_PHASE;
        }
    }
}

```

Fotos des Aufbaus



Schaltplan



Zustandsbasiertes Testen

1. Identifikation der Zustände

- Grüne Phase -> GREEN_PHASE
- Gelbe Phase -> YELLOW_PHASE
- Rote Phase -> RED_PHASE
- Rot-Gelbe-Phase -> RED_YELLOW_PHASE

2. Testfälle pro Zustand

Testfälle für Ampel LEDs

Zustand	Erwartetes Ergebnis	Ergebnis	Überprüfung	Bestanden (Ja/Nein)
Grüne Phase	LED für Autos • Rot = aus • Gelb = aus • Grün = an LED für Fußgänger • Rot = an • Grün = aus	LED für Autos • Rot = aus • Gelb = aus • Grün = an LED für Fußgänger • Rot = an • Grün = aus	Beobachtung, ob die richtigen LEDs leuchten.	Ja
Gelbe Phase	LED für Autos • Rot = aus • Gelb = an • Grün = aus LED für Fußgänger • Rot = an • Grün = aus	LED für Autos • Rot = aus • Gelb = an • Grün = aus LED für Fußgänger • Rot = an • Grün = aus	Beobachtung, ob die richtigen LEDs leuchten.	Ja
Rote Phase 1 bis 12 Sekunden	LED für Autos • Rot = an • Gelb = aus • Grün = aus LED für Fußgänger • Rot = aus • Grün = an	LED für Autos • Rot = an • Gelb = aus • Grün = aus LED für Fußgänger • Rot = aus • Grün = an	Beobachtung, ob die richtigen LEDs leuchten.	Ja
Rote Phase Ab 13 bis 15 Sekunden	LED für Autos • Rot = an • Gelb = aus • Grün = aus LED für Fußgänger • Rot = an • Grün = aus	LED für Autos • Rot = an • Gelb = aus • Grün = aus LED für Fußgänger • Rot = an • Grün = aus	Beobachtung, ob die richtigen LEDs leuchten.	Ja

Zustand	Erwartetes Ergebnis	Ergebnis	Überprüfung	Bestanden (Ja/Nein)
Rot-Gelbe-Phase	LED für Autos • Rot = an • Gelb = an • Grün = aus	LED für Autos • Rot = an • Gelb = an • Grün = aus	Beobachtung, ob die richtigen LEDs leuchten.	Ja
	LED für Fußgänger • Rot = an • Grün = aus	LED für Fußgänger • Rot = an • Grün = aus		

Testfälle für Taster / Zustandsübergänge

Zustand	Beschreibung	Erwartetes Ergebnis	Ergebnis	Bestanden (Ja/Nein)
Grüne Phase Nach 10 Sekunden	Der Taster wird gedrückt	Nach 20 Sekunden Übergang in die gelbe Phase	Nach 20 Sekunden Übergang in die gelbe Phase	Ja
Grüne Phase Nach 50 Sekunden	Der Taster wird gedrückt	Nach 10 Sekunden Übergang in die gelbe Phase	Nach 10 Sekunden Übergang in die gelbe Phase	Ja
Gelbe Phase	(Das Drücken des Tasters hat keine Auswirkung)	Nach 2 Sekunden Übergang in die rote Phase	Nach 2 Sekunden Übergang in die rote Phase	Ja
rote Phase	(Das Drücken des Tasters hat keine Auswirkung)	Nach 15 Sekunden Übergang in die rot-gelbe Phase	Nach 15 Sekunden Übergang in die rot-gelbe Phase	Ja
Rot-Gelbe-Phase	(Das Drücken des Tasters hat keine Auswirkung)	Nach 3 Sekunden Übergang in die grüne Phase	Nach 3 Sekunden Übergang in die grüne Phase	Ja

Testfälle für Blitzer LED / HC-SR04 Ultrasonic Distance Sensor

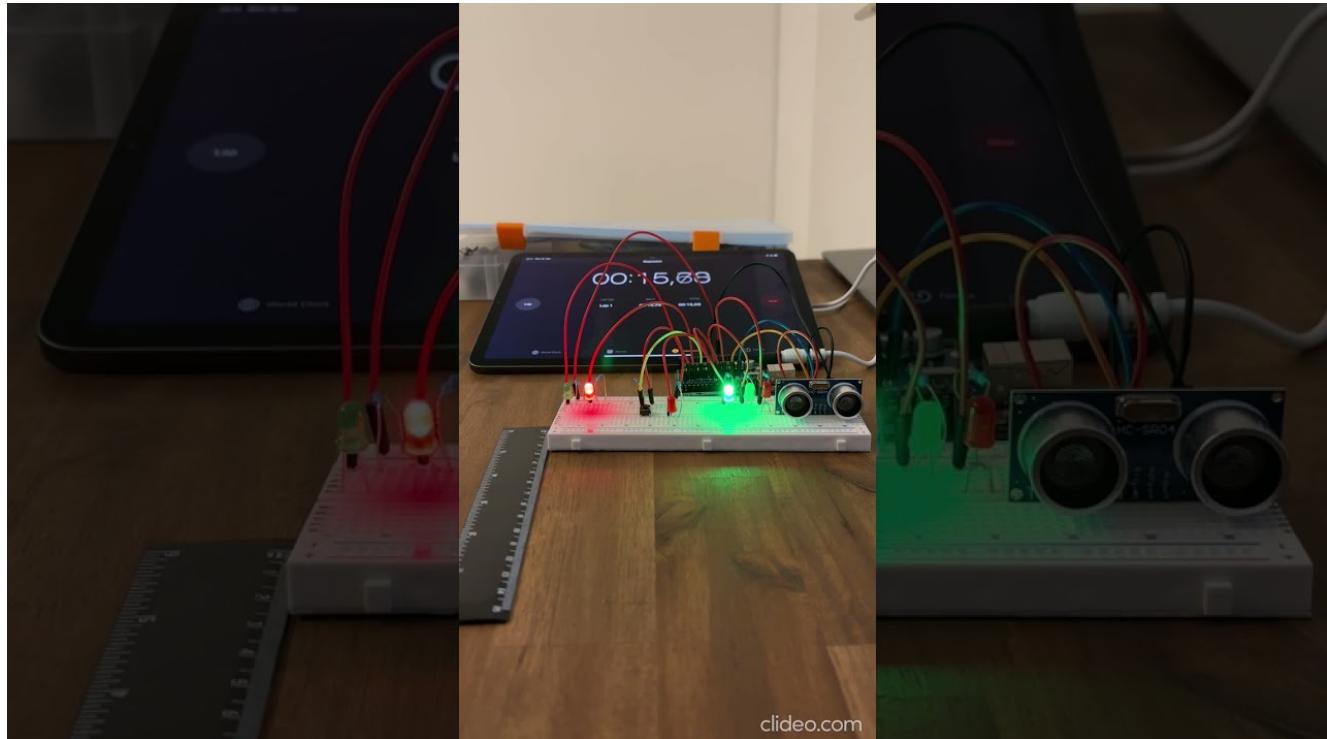
Zustand	Beschreibung	Erwartetes Ergebnis	Ergebnis	Bestanden (Ja/Nein)
Grüne Phase	Eine Hand wird ca. 5 cm vor den Sensor gehalten	Blitzer LED = aus	Blitzer LED = aus	Ja
Gelbe Phase	Eine Hand wird ca. 5 cm vor den Sensor gehalten	Blitzer LED = aus	Blitzer LED = aus	Ja

Zustand	Beschreibung	Erwartetes Ergebnis	Ergebnis	Bestanden (Ja/Nein)
Rote Phase	Eine Hand wird ca. 5 cm vor den Sensor gehalten	Blitzer LED schaltet in kurzen Abstände ein und aus	Blitzer LED schaltet in kurzen Abstände ein und aus	Ja
Rote Phase	Kein Hindernis in Reichweite des Sensors	Blitzer LED = aus	Blitzer LED = aus	Ja
Rot-Gelbe-Phase	Eine Hand wird ca. 5 cm vor den Sensor gehalten	Blitzer LED schaltet in kurzen Abstände ein und aus	Blitzer LED schaltet in kurzen Abstände ein und aus	Ja
Rot-Gelbe-Phase	Kein Hindernis in Reichweite des Sensors	Blitzer LED = aus	Blitzer LED = aus	Ja

3. Negativtests

Testfall	Beschreibung	Erwartetes Ergebnis	Ergebnis	Bestanden (Ja/Nein)
Ausfall des Distanzsensors	Der Distanzsensor ist nicht richtig angeschlossen	Kein blitzen während der rote Phase und der rot-gelbe Phase	Während der rote Phase und der rot-gelbe Phase wird dauerhaft geblitzt	Nein
Ausfall des Blitzer-LEDs	Blitzer-LED ist defekt oder nicht richtig angeschlossen	Kein blitzen während der rote Phase und der rot-gelbe Phase	Kein blitzen während der rote Phase und der rot-gelbe Phase	Ja

Live-Demo



Quellenangaben

Deckblatt Image: <https://copilot.microsoft.com/> ("Need an image for a traffic light controller Arduino project") [18.03.2024]

FSM: [https://de.wikipedia.org/wiki/Automat_\(Informatik\)](https://de.wikipedia.org/wiki/Automat_(Informatik)) [18.03.2024]

Aufgabenstellung: <https://mo6302.schule.hessen.de/mod/assign/view.php?id=43507> [18.03.2024]

Arduino Dokumentation: <https://www.arduino.cc/> [18.03.2024]

Taster Information: <https://chat.openai.com/c/c6c196ad-dc28-45d6-b30c-3dd03d3699ac> ("How can I check the state of a button in Arduino?") [13.03.2024]

Taster Funktion: https://www.youtube.com/watch?v=VPGRqML_v0w [13.03.2024]

C++ docstrings: <https://developer.lsst.io/cpp/api-docs.html> [13.03.2024]

Schaltplan und Tests: <https://wokwi.com/projects/39173942022555457> [18.03.2024]

Zustandsbasiertes Testen: <https://blog.milsystems.de/2012/03/der-zustandsbasierte-test-nach-istqb-in-der-praxis/> [18.03.2024]

Zustandsbasiertes Testen: https://de.wikipedia.org/wiki/Zustandsbezogener_Test [15.03.2024]

Versionverwaltung: <https://github.com/McLoving-81/ARDUINO-traffic-light> [18.03.2024]

Videoeditor: <https://clideo.com/> [18.03.2024]

Musik: <https://pixabay.com/de/music/search/genre/aufzugsmusik/> [18.03.2024]

Eigenständigkeitserklärung

Eidesstattliche Erklärung

Ich versichere an Eides statt durch meine eigene Unterschrift, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und alle Stellen, die wörtlich oder annähernd wörtlich aus Veröffentlichungen genommen sind, als solche kenntlich gemacht habe. Die Versicherung bezieht sich auch auf in der Arbeit gelieferte Zeichnungen, Skizzen, bildliche Darstellungen und dergleichen.

Datum: 19.03.2024

Unterschrift:

