RTT\_v3\_4x7

(RTT)

15.06.2022

smart Robotics

Table of Contents

[Terms of reference 3](#_Toc105954456)

[Component list 3](#_Toc105954457)

[Circuit 4](#_Toc105954458)

[Structure 5](#_Toc105954459)

[Source code with comments 6](#_Toc105954460)

# Terms of reference

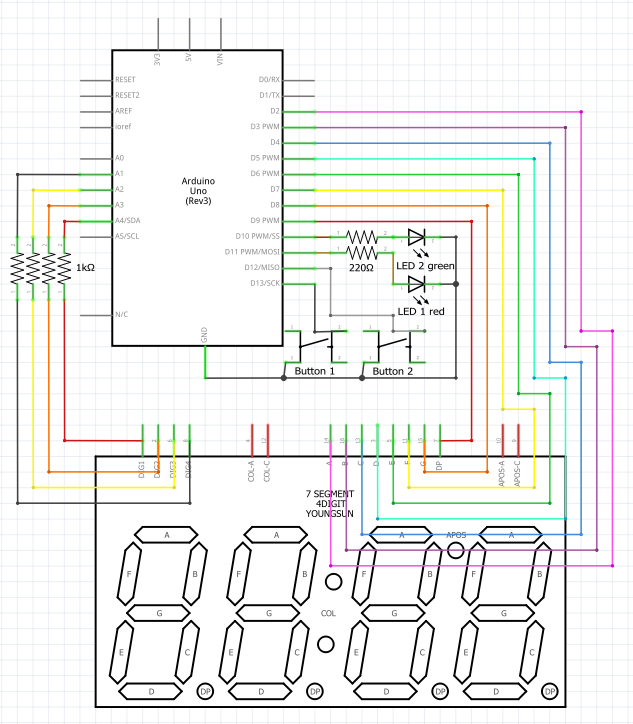
The task is to build and program a reaction time tester (RTT). The tester must consist out of two buttons and two LEDs. It should work like described below:

1. The green LED is on, and you must press the first button.
2. When the first button gets pressed the LED starts blinking fast and you have to wait for the red light.
3. As soon as the red LED lights up and the green LED goes out, the time is counted until the second button is pressed.
4. After the second button is pressed the red light goes out and you can read your reaction time on the display.

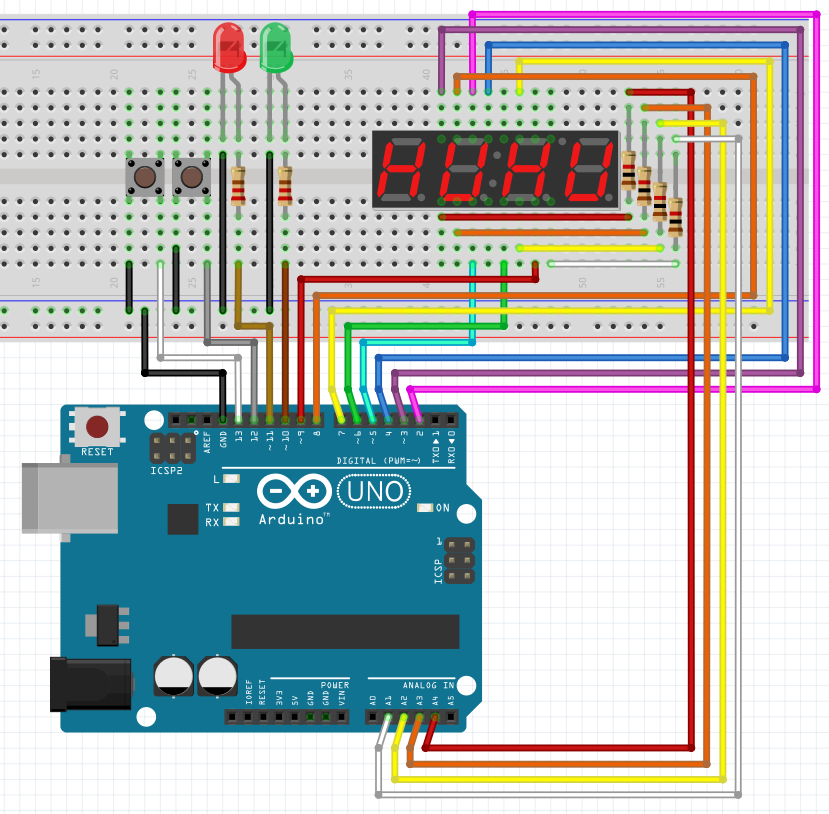
# Component list

|  |  |  |  |
| --- | --- | --- | --- |
| Quantity | Part | Symbol | Picture |
| 2 | 220Ω Resistor |  | E-Projects 100EP512220R 220 Ohm Resistors, 1/2 W, 5% (Pack of 100):  Amazon.com: Industrial & Scientific |
| 1 | Arduino Uno | Ein Bild, das Text, Antenne enthält.  Automatisch generierte Beschreibung | Arduino UNO Rev3 [A000066] : Amazon.de: Computer & Zubehör |
| 2 | Button | Schalter — Grundwissen Elektronik | Pts645 Series 6 Mm Tact Switches |
| 1 | green LED 5mm |  | Led verte diffusante 5mm |
| 1 | red LED 5mm |  | L-7113SURCK | Kingbright LED 645nm Rot 5 mm T-1 3/4 | Distrelec Deutschland |
| 1 | 4-digit 7-segment display |  | 7 Segment Display - 4 Bits - Ortak Katot - 0.56 inç Satın Al | Robo90 |

# Circuit



# Structure



# Source code with comments

#include <Bounce2.h> // including library Bounce2

#include <SevSeg.h> // including library SevSeg

SevSeg sevseg; //Instantiate a seven segment controller object

#define btn1 13 // button 1 yellow

#define btn2 12 // button 2 orange

class Led { // class Led

  int pin; // pin

  bool state; // state

  public:

  Led(int ledPin){ // constructor

    pin = ledPin; // pin

    pinMode(pin, OUTPUT); // pin mode

  } // end constructor

  void toggle(){ // toggle led

    state = !state; // state

    digitalWrite(pin, state); // set state for pin

  } // end toggle

  void speedToggle(int ms, int maxC){ // speed toggle

    int c; // counter

    while (c < maxC) // while counter is less than max counter

    {

      toggle(); // toggle led

      delay(ms); // delay

      c = c + random(ms); // counter

    } // end while

  } // end speed toggle

}; // end class Led

int gState = 1; // global state

long start; // global start time

long stop; // global stop time

float result; // global result

Bounce btn1Bouncer = Bounce(); // creating Bounce object for button 1

Bounce btn2Bouncer = Bounce(); // creating Bounce object for button 2

void setup(){ // setup function

  byte numDigits = 4; // number of digits

  byte digitPins[] = {A4, A3, A2, A1}; // digit pins

  byte segmentPins[] = {2, 3, 4, 5, 6, 7, 8, 9}; // segment pins

  bool resistorsOnSegments = false; // 'false' means resistors are on digit pins

  byte hardwareConfig = COMMON\_CATHODE; // See README.md for options

  bool updateWithDelays = false; // Default 'false' is Recommended

  bool leadingZeros = false; // Use 'true' if you'd like to keep the leading zeros

  bool disableDecPoint = false; // Use 'true' if your decimal point doesn't exist or isn't connected

  sevseg.begin(hardwareConfig, numDigits, digitPins, segmentPins, resistorsOnSegments, updateWithDelays, leadingZeros, disableDecPoint); // begin seven segment controller

  sevseg.setBrightness(50); // set brightness

  btn1Bouncer.attach(btn1, INPUT\_PULLUP); // attaching button 1 to Bounce object

  btn2Bouncer.attach(btn2, INPUT\_PULLUP); // attaching button 2 to Bounce object

  btn2Bouncer.interval(25); // setting interval for button 2

  btn1Bouncer.interval(25); // setting interval for button 1

}

Led led2(11); // creating Led object for led 2 red

Led led1(10); // creating Led object for led 1 green

void loop(){ // loop function

  btn1Bouncer.update(); // updating Bounce object for button 1

  btn2Bouncer.update(); // updating Bounce object for button 2

  switch(gState){ // switch statement for global state

    case 1: // case 1

      led1.toggle(); // toggle led 1

      gState = 2; // global state

      break; // break

    case 2: // case 2

      if (btn1Bouncer.fell()){ // if button 1 fell

        gState = 3; // global state

      }

      break; // break

    case 3: // case 3

      gState = 4; // global state

      break; // break

    case 4: // case 4

      led1.speedToggle(300, random(5\*1000)); // speed toggle led 1

      led2.toggle(); // toggle led 2

      led1.toggle(); // toggle led 1

      start = millis(); // start time

      gState = 5; // global state

      break; // break

    case 5: // case 5

      if (btn2Bouncer.fell()){ // if button 2 fell

        stop = millis(); // stop time

        led2.toggle(); // toggle led 2

        led1.toggle(); // toggle led 1

        result = stop - start; // result

        if (result < 5000) { // if result is less than 5000

          sevseg.setNumberF(result / 1000, 2); // set number for seven segment controller

        }

        else if (result >= 5000) { // if result is greater than or equal to 5000

          sevseg.setChars("noob"); // set number to noob

        }

        led1.toggle(); // toggle led 1

      }

      sevseg.refreshDisplay(); // refresh display

      break; // break

    default: // default case

      sevseg.setChars("FAIL"); // set characters

      sevseg.refreshDisplay(); // refresh display

      break; // break

  } // end switch statement

} // end loop function