

Zunächst soll eine Stoppuhr programmiert werden, die nach jedem Tasterdruck die Stoppuhr neu startet und anschließend das Ergebnis anzeigt. Hier sollen die Sekunden noch in Minuten umgerechnet werden, wenn jeweils 60 Sekunden überschritten werden.

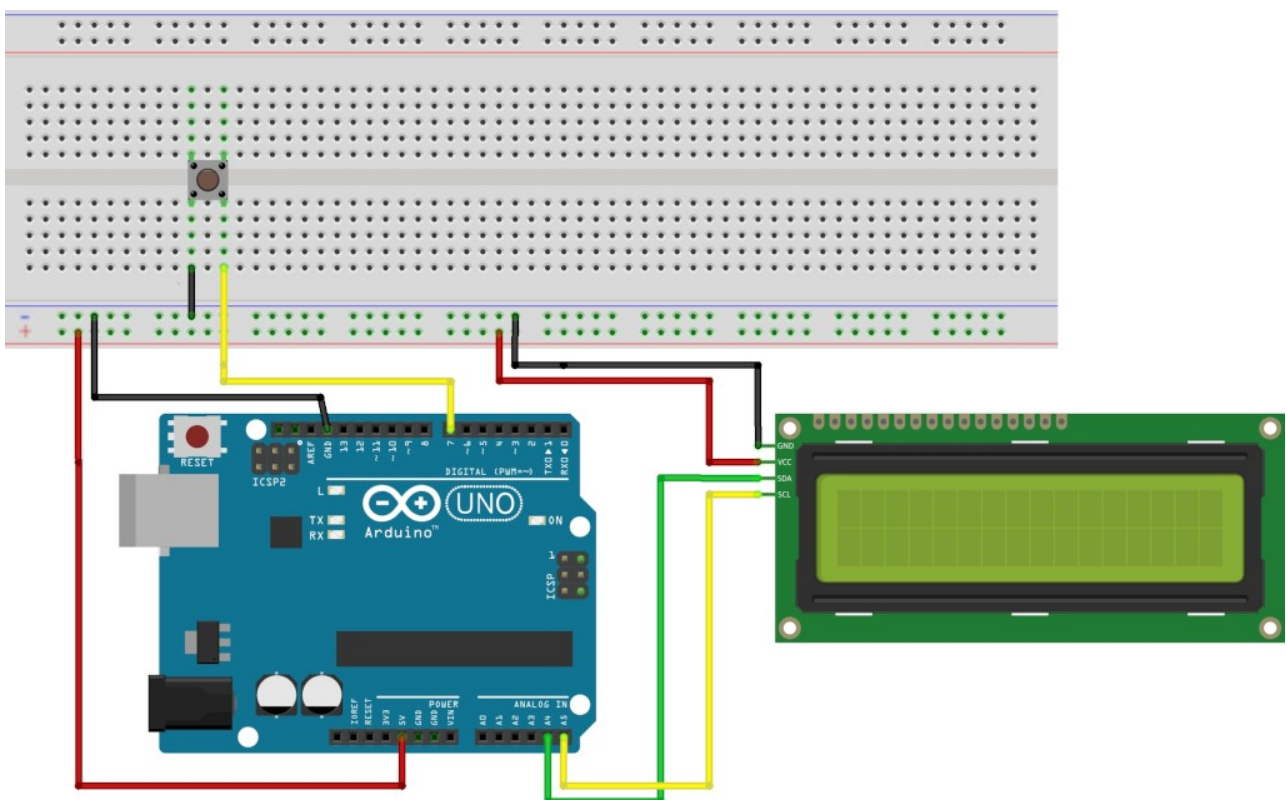
So soll es aussehen:



Benötigte Bauteile:

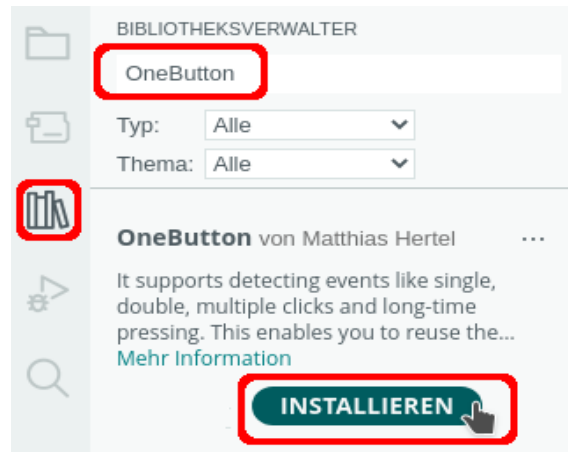
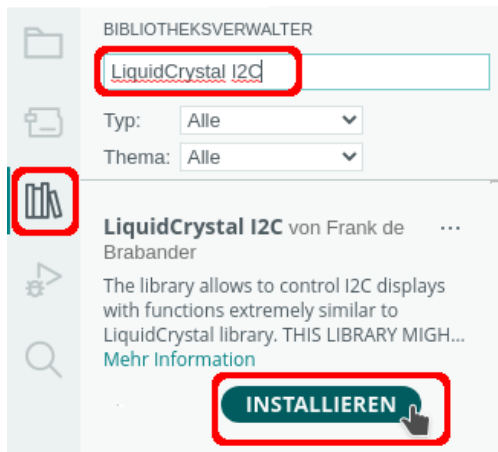
- ➔ LCD 1602
- ➔ Taster
- ➔ Leitungsdrähte

Baue die Schaltung auf.



fritzing

Benötigte Bibliotheken:



Binde die benötigten Bibliotheken ein und definiere die Variablen:

```
# include <LiquidCrystal_I2C.h>
# include <OneButton.h>

int TASTER = 7;

// Name des Tasters
OneButton NameTaster(TASTER, true);

// Name des LCDs
LiquidCrystal_I2C lcd(0x27, 20, 4);

// Variablen der Zeit
float StartZeit;
float VerstricheneZeit;
float Sekunden;
bool Anzeige = true;

// TasterStatus sorgt für Start/Neustart
bool TasterStatus = true;
```

Der setup-Teil startet das LCD und macht die benötigten Einstellungen für die Bibliothek OneButton.

```
void setup()
{
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Taste -> Start");
  lcd.setCursor(0, 1);
  lcd.print("-----");
}
```

```
// Aktionen dem Modus des Tasters zuordnen
NameTaster.attachClick(einKlick);

/*
  Anzahl der Millisekunden festlegen
  Standardwerte gesetzt:
  PressTicks: 1000
  ClickTicks: 600
  DebounceTicks: 50
  wenn die Standardwerte gesetzt werden sollen
  können die nächsten Zeilen auskommentiert werden
*/
NameTaster.setPressTicks(1000);
NameTaster.setClickTicks(500);
NameTaster.setDebounceTicks(50);
}
```

Der loop-Teil sorgt lediglich dafür, dass der Taster abgefragt wird:

```
void loop()
{
  // Taster alle 10 Millisekunden abfragen
  NameTaster.tick();
  delay(10);
}
```

Die Methode einKlick() misst die Zeit und sorgt für die Ausgabe auf dem LCD:

```
void einKlick()
{
  if (TasterStatus)
  {
    StartZeit = millis();
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Taste - Stopp");
    lcd.setCursor(0, 2);
  }

  else
  {
    // Zeit berechnen
    float Sekunden;
    VerstricheneZeit = millis() - StartZeit;

    Sekunden = VerstricheneZeit / 1000;
    String GesamtSekunden = String(Sekunden);

    // . durch , ersetzen
    GesamtSekunden.replace(".", ",");
  }
}
```

```
// Ausgabe im Seriellen Monitor
Serial.println("Sekunden insgesamt: " + GesamtSekunden + " Sekunden");

// Minuten berechnen
int Minute = int(Sekunden) / 60;

// nur Ausgabe der Minuten wenn Minute > 0
if (Minute > 0)
{
    // Ausgabe verschönern, wenn Minute > 1 -> Ausgabe "Minuten"
    // "Minute"
    if (Minute > 1)
    {
        lcd.setCursor(0, 2);
        lcd.print(String(Minute) + " Minuten ");
    }

    else
    {
        lcd.setCursor(0, 2);
        lcd.print(String(Minute) + " Minute ");
    }
}

// von Sekunden Anzahl der Minuten abziehen
Sekunden = Sekunden - Minute * 60;

// Sekunden in String umwandeln
// damit . durch , ersetzt werden kann
String AnzahlSekunden = String(Sekunden);

// . durch , ersetzen
AnzahlSekunden.replace(".", ",");

// Ausgabe LCD
lcd.setCursor(0, 0);
lcd.print("Neustart -> Taste");
lcd.setCursor(0, 3);
lcd.print(AnzahlSekunden + " Sekunden");
}

// Tasterstatus umkehren
TasterStatus = !TasterStatus;
}
```

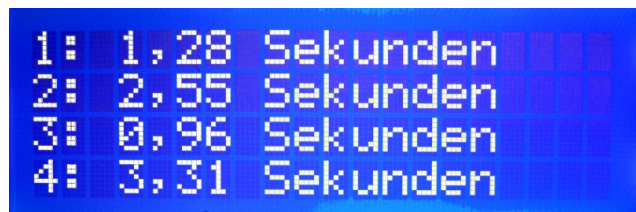
Die Stoppuhr soll jetzt erweitert werden. Ein Druck auf den Taster startet die Stoppuhr, der nächste Druck stoppt sie. Es sollen vier Runden erfasst werden.

Ein Doppelklick auf den Taster zeigt die Ergebnisse der bisherigen Runden an, ein langer Druck startet die Stoppuhr wieder bei Runde eins.



Eine Stoppuhr für den Seriellen Monitor findest du [hier](#).

So soll es aussehen:



Ergänze die Variablen:

```
# include <LiquidCrystal_I2C.h>
# include <OneButton.h>

int TASTER = 7;

// Name des Tasters
OneButton NameTaster(TASTER, true);

// Name des LCDs
LiquidCrystal_I2C lcd(0x27, 20, 4);

// Variablen der Zeit
float StartZeit;
float VerstricheneZeit;
float Sekunden;

int AnzahlRunden = 1;

const int RundenMax = 5;

// Array Runden
String RundenZeit[RundenMax];

// TasterStatus sorgt für Start/Neustart
bool TasterStatus = true;

bool Neustart = false;
```

Im setup-Teil müssen die Aktionen für den Doppelklick und den langen Tasterdruck zusätzlich definiert werden:

```
void setup()
{
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Taste -> Start");
  lcd.setCursor(0, 1);
  lcd.print("-----");

  // Aktionen dem Modus des Tasters zuordnen
  NameTaster.attachClick(einKlick);
  NameTaster.attachDoubleClick(DoppelKlick);
  NameTaster.attachLongPressStop(langerDruckStopp);

  /*
    Anzahl der Millisekunden festlegen
    Standardwerte gesetzt:
    PressTicks: 1000
    ClickTicks: 600
    DebounceTicks: 50
    wenn die Standardwerte gesetzt werden sollen
    können die nächsten Zeilen auskommentiert werden
  */
  NameTaster.setPressTicks(1000);
  NameTaster.setClickTicks(500);
  NameTaster.setDebounceTicks(50);
}
```

Im loop-Teil gibt es keine Änderungen.

Die Methode einKlick() wird ergänzt. Beachte die Kommentare.

```
void einKlick()
{
  if (TasterStatus)
  {
    if (Neustart)
    {
      AnzahlRunden = 1;

      // String leeren
      for (int i = 1; i < RundenMax; i ++)
      {
        RundenZeit[i] = "";
      }
      Neustart = false;
    }
    StartZeit = millis();
    Serial.println("Runde: " + String(AnzahlRunden));
    lcd.clear();
  }
}
```

```

    lcd.setCursor(0, 0);
    lcd.print("Runde: " + String(AnzahlRunden));
    lcd.setCursor(0, 1);
    lcd.print("Taste -> Stopp");
}

else
{
    // Zeit berechnen
    float Sekunden;
    VerstricheneZeit = millis() - StartZeit;
    Sekunden = VerstricheneZeit / 1000;

    // Sekunden in String umwandeln
    // damit . durch , ersetzt werden kann
    String AnzahlSekunden = String(Sekunden);

    // . durch , ersetzen
    AnzahlSekunden.replace(".", ",");

    // Ausgabe LCD
    lcd.clear();
    lcd.print("Runde: " + String(AnzahlRunden));

    lcd.setCursor(0, 1);
    lcd.print("Weiter -> Taste");

    lcd.setCursor(0, 3);
    lcd.print(AnzahlSekunden + " Sekunden");

    if (AnzahlRunden == RundenMax - 1)
    {
        RundenZeit[AnzahlRunden] = AnzahlSekunden;
        lcd.clear();
        Neustart = true;
        DoppelKlick();
    }
    else
    {
        RundenZeit[AnzahlRunden] = AnzahlSekunden;
        AnzahlRunden ++;
    }
}

// Tasterstatus umkehren
TasterStatus = !TasterStatus;
}

```

Die Methode `DoppelKlick()` zeigt das Ergebnis der Runden an, die Methode `langerDruckStopp()` sorgt für den Neustart.

```
void DoppelKlick()
{
    lcd.clear();
    int Zeile = 0;
    for (int i = 1; i <= RundenMax - 1; i++)
    {
        lcd.setCursor(0, Zeile);
        if (RundenZeit[i] != "") lcd.print(String(i) + ": " + RundenZeit[i] + "
        Sekunden");
        Zeile++;

        // Serieller Monitor
        if (RundenZeit[i] != "") Serial.println(String(i) + ": " + RundenZeit[i] +
        " Sekunden");
    }
}

void langerDruckStopp()
{
    AnzahlRunden = 1;
    TasterStatus = true;
    Neustart = true;
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Neustart: Runde 1");
}
```