Weather Clock

*Echtzeit Datendarstellung auf dem Display

1st Imron Gamidli
Internationale Ingenieurwissenschaften
Hochschule Fulda
Fulda, Deutschland
imron.gamidli@lt.hs-fulda.de

2nd Hussain Ali
Internationale Ingenieurwissenschaften
Hochschule Fulda
Fulda, Deutschland
husain.ali@lt.hs-fulda.de

***Abstract*—**
***Index Terms*—dht22, arduino, uno,**

I. Einführung

Dieses Projekt geht um ein Weather-Clock, dass es verschiedene Funktionen haben. Die Idee ist, eine Wetteruhr mit mehreren Funktionen zu haben, die verschiedene Merkmale von Wetter und Zeit messen und anzeigen kann. Die Funktionen werden flexibel und Portal sein. Man konnte die Funktion der Uhrzeit und des Datums ändern und auch die Eigenschaften des Wetters sehen. Die Motivation hinter diesem Projekt ist die Einsparung von Strom sowie die Berücksichtigung des Bildschirms durch einen Bewegungssensor, der neben seinen Funktionalitäten eine gute Initiative zum Energiesparen sein kann.

Wir haben viel über die Wetteruhr recherchiert, aber diese Energiesparfunktion hat gefehlt. Wir dachten, es wäre eine gute Idee, es zu haben. Wenn man sich im Raum oder in der Nähe des Bildschirms nicht befindet, sollte die Uhr oder das Display nicht eingeschaltet sein, wenn sich im Raum etwas bewegt, wird es wieder angezeigt. In der Zukunft kann man die Merkmale und Funktionen dieser Uhr erweitern und aktualisieren. Dies kann auch als Wecker verwendet werden. Da wir einen Bewegungssensor verwenden, um die Bewegung und das Drehen der Displayhelligkeit zu erfassen, kann man ihn auch als Alarm verwenden, um zu benachrichtigen, dass jemand kommt oder geht, insbesondere in öffentlichen Geschäften.

Es misst und zeigt die Uhrzeit, die Temperatur, die F¬euchtigkeit, den Luftdruck und die Höhenlage. Die Funktionen der Uhrzeit sind interaktiv, das heißt, kann man durch drei Knöpfen das Datum und Uhr andern und speichern. Die Darstellung von Uhr und Temperatur werden auf den zwei Displays gezeigt. Die beide sind 2x16 I2C Displays. Wir haben drei Sensoren, BMP280, DHT22, PIR verwendend. Der BMP280 misst der Luftdruck und die Höhenlage, der DHT22 misst die Temperatur, die Feuchtigkeit. Durch PIR Bewegung Sensor wird überprüft, wenn keine Bewegung in dem Raum ist, wird dann die Helligkeit von Displays ausgeschalten.

Außerdem wird RTC (Real Time Clock) benutzt, können wir dadurch die Echtzeit Uhr und Datum auf dem Display zeigen. Zudem RTC gibt es drei Knöpfen, die Funktionen von RTC andern und speichern lassen.

Zum Protokoll verwenden wir I2C, mit dem kann man mehrere Slaves mit einem einzigen Master (wie SPI) verbinden und auch kann man mehrere Master haben, die einzelne oder mehrere Slaves steuern. Es ist sehr nützlich, wenn man mehr als ein Mikrocontroller Daten auf einer einzigen Speicherkarte protokolliert oder Text auf einem einzigen LCD anzeigt.

II. Protokoll

Das I2C-Protokoll ist das Serial-Communication-Protocol, die für langsame Mikrocontrollern Geräte, z. B. EEPROM (Electrically Erasable Programmable Read-only Memory), ADC (Analog To Digital Converter) und RTC (Real Time Clock) verwendet werden. Der I2C ist ein Zweidraht-Kommunikationsprotokoll, dass für die Kommunikation nur zwei Kabel verwendet. Dabei wird ein Draht für die Daten SDA (Serial data line) und der andere Draht für die Clock SCL (Serial clock line) verwendet. Beim I2C sind die Kommunikation zwischen beide Buse bidirektional, das heißt, dass der Master die Daten vom Slave schicken und empfangen kann. Tatsächlich wird der Taktbus durch den Master kontrolliert. Jedoch kann der Slave manchmal das Clock-signal unterdrücken.

A. Schnittstelle

Der I2C benutzen zwei Leitungen SCL und SDA, welche bidirektionale Buse sind. Die bidirektionale Buse Linien werden durch Master und Slave unter Verbindung von Open-Drain-Ausgangstufen und einem Pull-up-Widerstand (Rp) implementiert, die mit einer positiven Versorgungsspannung verbunden sind.

Der Pull-up-Widerstandswert hängt vom Systemaufbau, von der Kapazität der Schaltung oder der Kabel- und Bustaktfrequenz, ab. Logischerweise kann der Wert für den Pull-up-Widerstandswert 10kohm sein und die Kapazitäten-Lasten von SDA und SCL Leitung müssen gleich sein. Die Kapazitäten-Lasten-Leitung soll nicht länger als 30cm sein.

B. Adressen

Bei der grundlegenden Kommunikation des I2Cs werden die Übertragung von 8 Bits oder Bytes verwendet. Jedes Slave des I2C Geräts hat eine 7-Bit Adresse, die auf dem Bus eindeutig sein sollen. Jedoch braucht das Master-Gerät

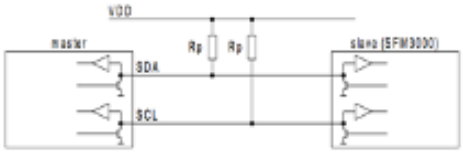

Fig. 1. Beispiel von SDA und SCL mit bi-direktionalen Kommunikation und externerem Pull-up-Widerstand (Rp).

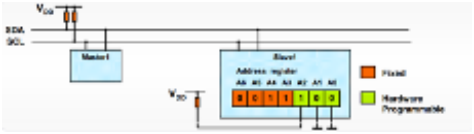

Fig. 2. Beispiel von 7-Stillegen Binärwert Adresse

keine Adresse, weil das Clock (über SCL) generiert und an die individuelle Slave-Geräte adressiert. Tatsächlich wird die I2C-Adresse durch einen 7-stelligen Binärwert dargestellt. Standardmäßig ist die Adresse auf 64 (1000000b) eingestellt. Sie folgt immer durch ein Schreib-bit (0) und ein Lese-bit (1). Wenn Bit 0 auf 1 gesetzt wird, liest das Master-Gerät vom Slave-Gerät.

C. Kommunikation

Der I2C ist ein 8-Bit-Kommunkationsprotokoll. Das I2C-Protokoll hat ein Start und ein Stopp Bedingungen zu Beginn und am Ende einer Übertragung. Beim I2C wird durch ACK- (Acknowledgement) oder NACK- (Not Acknowledgement) nach jedem Byte bestätigt.

- Die Kommunikation muss mit START angefangen werden : START-Bedingung
- Dem Start Bit wird immer durch Adresse von Slave gefolgt.
- Die Adresse von Slave werden durch READ und NOT WRITE Bit gefolgt.
- Das Empfänger-Gerät (entweder Master oder Slave) müssen ein ACKNOWLEDGE Bit schicken.
- Die Kommunikation müssen mit STOP anfangen werden : STOP-Bedingung

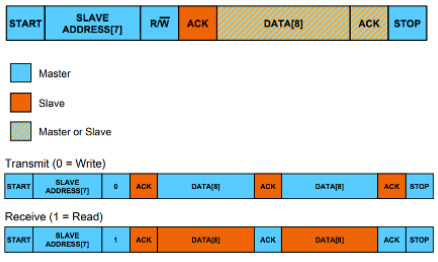

Fig. 3. Beispiel von Kommutation von SRART und STOP durch Write(0) und Read(1) Übertragung.

Fig. 4. Beispiel von START und STOP – Bedingungen

D. START und STOP – Bedingungen

1) START Bedingung: : Ein Übergang von einer hohen zu einer niedrigen SDA-Leitung, wird als START-Bedingung bezeichnet, da die SCL-Leitung hoch ist. Standardmäßig ist der Status von SDA und SCL Leitung hoch. Die Kommunikation auf Leitung wird von dem Master angefangen. Die START- Bedingung wird immer vom Master bestätigt und der I2C Bus wird nach der Bestätigung des START-Bits als besetzt betrachtet.

2) STOP Bedingung: : Eine Übergang von niedriger zu hoher SDA-Leitung, während die SCL-Leitung hoch ist, wird als STOP-Bedingung bezeichnet. Die STOP–Bedingung wird immer vom Master aktiviert, um die Kommunikation zu stoppen. Nach der Aktivierung des STOP Bits wird den I2C- Bus frei.

- Start Bedingung: ein Übergang von HOCH nach NIEDRIG auf der
- Stop Bedingung: ein Übergang von NIEDRIG nach HOCH auf der SDA-Leitung, weil SCL HOCH ist

E. Read /Write Bit

Der Adresse-frame hat einzelne Bits am Ende und sagt dem Slave Bescheid, ob der Master ein Datei davon empfangen und schreiben möchte. Wenn der Master die Daten an den Slave schicken möchte, ist das Read/Write Bit ein Niederspannungspegel. Wenn der Master eine Datei von dem Slave fordert, ist das ein Hochspannungspegel.

F. Datentransfer

Die Daten werden in 8-Bit Paketen (Bytes) auf dem I2C Bus übertragt. Es gibt keine Begrenzung bei der Anzahl der Bytes, aber jedes Byte muss vom Bestätigungsbit (Acknowledge bit) gefolgt werden. Das Bit signalisiert, ob das Gerät bereit ist, mit dem nächsten Byte fortzufahren. Der Master muss die Clock- Pulse für alle Datenbits und Bestätigungsbit generieren. Wenn die Pulse des Slave -Geräts nicht die Übertragung bestätigen, bedeutet das, dass es keine weiteren Daten gibt oder die Übertragung nicht bereit ist. Dann muss das Master-Gerät entweder eine Stopbedingung oder eine wiederholte Startbedingung erzeugen

- Jedes Byte soll von einem Bestätigungsbit (acknowledge bit) gefolgt werden.
- Die Anzahl der übertragenen Datenbytes (data bytes) pro Übertragung ist nicht eingeschränkt.
- Wenn ein Slave kein weiteren Datenbyte empfangen oder senden kann, kann er die Taktleitung SCL LOW (Clock strechting) halten, um den Master in einen Wartezustand zu zwingen.

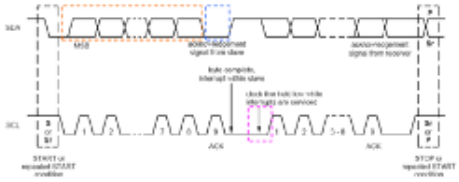

Fig. 5. Beispiel von Datentransfer

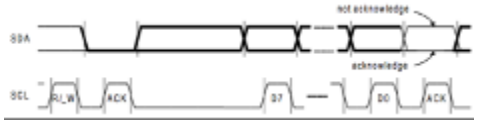

Fig. 6. Beispiel von Acknowdelge /Not Acknowledge

G. Acknowledge (ACK) / Not Acknowledge (NACK)

Jedes übertragene Byte wird auf dem I2C Bus von einer Acknowledge-Bedingung von dem Empfänger gefolgt. Das heißt, dass nachdem der Master SCL nach NIEDRIG gezogen hat, um die Übertragung des 8. Bits zu ergänzen, wird der SDA während der 9.Bitzeit nach NIEDRIG von dem Empfänger gezogen. Wenn nach der Übertragung des 8.Bits vom Empfänger der SDA Leitung nicht nach NIEDRIG gezogen wird, bezeichnet man das als Not-Acknowledge (NACK) Bedingung. Das Schaubild zeigt, dass jedes Byte, das vom Empfänger generiert wird, von einem Acknowledge or Not Achnowledge gefolgt wird. Außerdem zeigen die dick gedruckten Linien, dass der Sensor die SDA-Linie kontrolliert.

III. DHT22 Sensor

A. Beschreibung

Der DHT22 ist ein einfacher, preiswerter digitaler Temperatur- und Feuchtigkeitssensor. Er verwendet einen kapazitiven Feuchtigkeitssensor und einen Thermistor, um die Umgebungsluft zu messen, und spuckt ein digitales Signal auf dem Datenpin aus (keine analogen Eingangspins erforderlich). Er ist recht einfach zu bedienen, erfordert aber ein sorgfältiges Timing bei der Datenerfassung. Der einzige wirkliche Nachteil dieses Sensors ist, dass man nur alle 2 Sekunden neue Daten von ihm erhalten kann, so dass bei Verwendung unserer Bibliothek die Sensormesswerte bis zu 2 Sekunden alt sein können.

Verbinden Sie einfach den ersten Pin auf der linken Seite mit 3-5V Strom, den zweiten Pin mit Ihrem Dateneingangspin und den ganz rechten Pin mit Masse. Obwohl der Sensor ein einziges Kabel zum Senden von Daten verwendet, ist er nicht mit Dallas One Wire kompatibel! Wenn Sie mehrere Sensoren wünschen, muss jeder seinen eigenen Datenpin haben.

Im Vergleich zum DHT11 ist dieser Sensor präziser, genauer und funktioniert in einem größeren Temperatur-/Feuchtigkeitsbereich, ist aber auch größer und teurer.

Wird mit einem 4.7K - 10K Widerstand geliefert, den Sie als Pullup vom Daten-Pin zu VCC verwenden wollen.

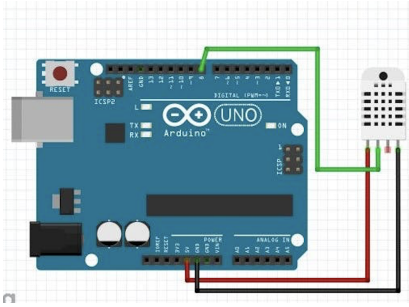

Fig. 7. DHT22 Sensor

B. Technische Daten

- Geringe Kosten
- 3 bis 5V Stromversorgung und E/A
- Maximal 2,5 mA Stromverbrauch während der Umwandlung (während der Datenabfrage)
- Gut geeignet für 0-100
- Gut geeignet für Temperaturmessungen von -40 bis 80°C mit einer Genauigkeit von ±0,5°C
- Abtastrate nicht mehr als 0,5 Hz (einmal alle 2 Sekunden)
- Gehäusegröße 27mm x 59mm x 13,5mm (1,05” x 2,32” x 0,53”)
- 4 Stifte, 0,1” Abstand
- Gewicht (nur der DHT22): 2,4g

Es ist ziemlich einfach, die DHT-Sensoren anzuschließen. Sie haben vier Stifte VCC - roter Draht Schließen Sie an 3,3 - 5V Strom an. Manchmal reicht die 3,3-V-Spannung nicht aus, in diesem Fall versuchen Sie es mit einer 5-V-Spannung. Datenausgang - weißes oder gelbes Kabel Nicht angeschlossen Masse - schwarzes Kabel Ignorieren Sie einfach Pin 3, er wird nicht verwendet. Dieses Diagramm zeigt, wie wir für die Testskizze anschließen werden. Schließen Sie Daten an Pin 2 an, Sie können dies später auf jeden beliebigen Pin ändern.

C. Implementierung im Projekt

DHT22 war nützlich für unseres Projekt. Mithilfe diesen Sensor, bekommen wir Temperatur und Feuchtigkeit Daten. Damit es möglich wird, Sensordaten zu lesen, bestimmte Bibliothek muss installiert und importiert soll. Die Bibliothek für DHT22 heißt *DHT sensor library* und installiert wurde vom https://github.com/adafruit/DHT-sensor-library . Nachdem wir die Bibliothek installiert haben, haben wir diese importiert.

```
#include <DHT.h>;
```

Diese Bibliothek ermöglicht uns eine DHT Instanz initialisieren. Hier man sieht wir haben PIN 13 für DHT Kommunikation. Danach haben wir DHT22 Instanz erstellt.

```
#define DHTPIN 13 // what pin we're connected to
#define DHTTYPE DHT22 // DHT 22
DHT dht(DHTPIN, DHTTYPE); // Initialize DHT
```

Wir werden später Daten aus dem Sensor in Variablen speichern. Dafür haben wir zwei Variablen, hum für Feuchtigkeit und temp für Temperatur deklariert.

```
float hum;  //Stores humidity value
float temp; //Stores temperature value
```

In setup() Funktion haben wir die DHT Instanz gestartet. Das heißt, die Instanz läuft und wir können jederzeit Feuchtigkeit und Temperaturdaten anfragen.

```
void setup()
{
  dht.begin();
}
```

In der Schleife loop() das Programm liest Feuchtigkeitswert mit readHumidity() und speichert in hum variable. Auf die gleiche Weise Temperatur wird mit readTemperature() gelesen und in temp gespeichert.

```
void loop()
{
  //Read data and store it to variables hum and temp
  hum = dht.readHumidity();
  temp = dht.readTemperature();

}
```

Letzte Aufgabe ist auf dem Display darzustellen, das vorher mit entsprechender Adresse initialisiert wurde. Wir haben zwei Displays, für DHT Daten haben wir erste Reihe des erstens Display benutzt.

```
  void loop()
  {
    //Read data and store it to variables hum and
    temp
    hum = dht.readHumidity();
    temp = dht.readTemperature();

    lcd1.setCursor(0, 0);
    lcd1.print("tmp:");
    lcd1.print(temp);
    lcd1.setCursor(6, 0);
    lcd1.print((char)223);
    lcd1.print("C hum:");
    lcd1.print(hum);
    lcd1.setCursor(15, 0);
    lcd1.print("%");

  }
```

IV. PIR Motion Sensor

A. Überblick

PIR-Sensoren ermöglichen die Erfassung von Bewegungen und werden fast immer verwendet, um festzustellen, ob sich ein Mensch in den Sensorbereich hinein- oder herausbewegt hat. Sie sind klein, preiswert, stromsparend, einfach zu bedienen und verschleißen nicht. Aus diesem Grund werden sie häufig in Geräten und Vorrichtungen in Haushalten und Unternehmen eingesetzt.

PIR-Sensoren bestehen im Wesentlichen aus einem pyroelektrischen Sensor (den Sie unten als runde Metalldose mit einem rechteckigen Kristall in der Mitte sehen können), der die Stärke der Infrarotstrahlung erkennen kann. Alles gibt eine schwache Strahlung ab, und je heißer etwas ist, desto mehr Strahlung wird freigesetzt. Der Sensor in einem Bewegungsmelder ist eigentlich in zwei Hälften geteilt. Der Grund dafür ist, dass wir eine Bewegung (Veränderung) und

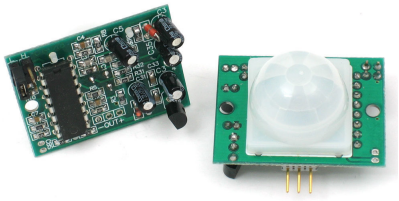

Fig. 8. PIR Sensor

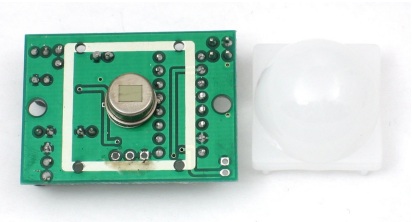

Fig. 9. PIR Sensor 1

nicht die durchschnittliche IR-Stärke erkennen wollen. Die beiden Hälften sind so verdrahtet, dass sie sich gegenseitig aufheben. Wenn eine Hälfte mehr oder weniger IR-Strahlung sieht als die andere, schwankt der Ausgang nach oben oder unten.

Zusammen mit dem pyroelektrischen Sensor gibt es eine Reihe von unterstützenden Schaltungen, Widerständen und Kondensatoren. Es scheint, dass die meisten kleinen Bastler-Sensoren den BISS0001 verwenden, zweifellos ein sehr preiswerter Chip. Dieser Chip nimmt das Ausgangssignal des Sensors auf und verarbeitet es ein wenig, um einen digitalen Ausgangsimpuls aus dem analogen Sensor zu erzeugen.

Im Vergleich zu älteren PIRs haben die neuen PIRs mehr einstellbare Einstellungen und sind mit einer Stiftleiste ausgestattet, die in den 3-poligen Masse-/Ausgangs-/Stromanschlüssen installiert ist.

Für viele einfache Projekte oder Produkte, bei denen erkannt werden muss, ob eine Person den Bereich verlassen oder betreten hat oder sich ihm genähert hat, sind PIR-Sensoren hervorragend geeignet. Sie sind stromsparend und kostengünstig,

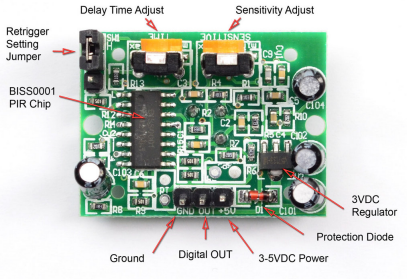

Fig. 10. PIR Sensor Einstellungen

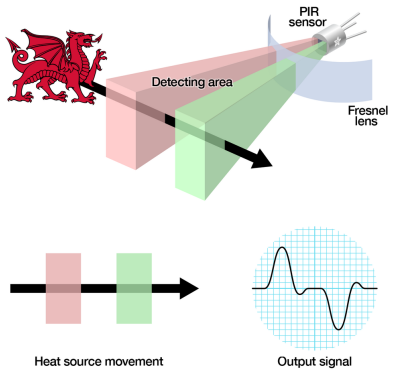

Fig. 11. PIR Sensor Funktionsweise

ziemlich robust, haben einen großen Linsenbereich und sind einfach zu bedienen. PIR-Sensoren zeigen nicht an, wie viele Personen sich in der Nähe aufhalten oder wie nah sie am Sensor sind. Die Linse ist oft auf einen bestimmten Bereich und eine bestimmte Entfernung festgelegt (obwohl sie irgendwo gehackt werden kann), und manchmal werden sie auch von Haustieren ausgelöst.

B. *Wie PIRs funktionieren*

PIR-Sensoren sind komplizierter als viele der anderen Sensoren, die in diesen Tutorials erklärt werden (wie Fotozellen, FSRs und Kippschalter), weil es mehrere Variablen gibt, die den Eingang und Ausgang des Sensors beeinflussen. Um zu erklären, wie ein grundlegender Sensor funktioniert, verwenden wir dieses schöne Diagramm

Der PIR-Sensor selbst hat zwei Schlitze. Jeder Schlitz besteht aus einem speziellen Material, das für IR empfindlich ist. Die Linse, die hier verwendet wird, macht nicht wirklich viel, und so sehen wir, dass die beiden Schlitze über eine gewisse Entfernung hinaus ”sehen” können (im Grunde die Empfindlichkeit des Sensors). Wenn sich der Sensor im Leerlauf befindet, erkennen beide Schlitze die gleiche Menge an IR, nämlich die vom Raum, den Wänden oder der Außenumgebung abgestrahlte Menge. Wenn ein warmer Körper, z. B. ein Mensch oder ein Tier, vorbeikommt, fängt er zunächst eine Hälfte des PIR-Sensors ab, was eine positive Differenzänderung zwischen den beiden Hälften bewirkt. Wenn der warme Körper den Erfassungsbereich verlässt, geschieht das Gegenteil, wobei der Sensor eine negative Differenzänderung erzeugt. Diese Änderungsimpulse sind das, was erkannt wird.

C. *Implementierung im Projekt*

Wir haben Pin 2 im Arduino für PIR Sensor benutzt und starten wir das Programm mit keine Bewegung. Für Lesen von PIN status, haben wir variable deklariert heißt val.

```
int inputPin = 2; // choose the input pin (for PIR
    sensor)
int pirState = LOW; // we start, assuming no motion
    detected
int val = 0;  // variable for reading the pin status
```

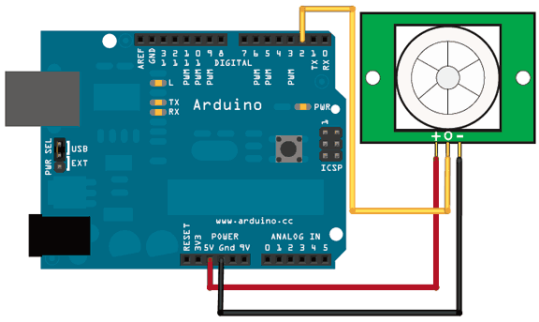

Fig. 12. PIR Sensor Funktionsweise

Im Setup deklarieren wir inputPin (pin 2) als INPUT pin. Das ermöglicht Sensor zu lesen.

```
void setup()
{
  pinMode(inputPin, INPUT); // declare sensor as
    input
}
```

Im Loop passiert vieles. In jede Iteration wird inputPin digital gelesen und zum val gespeichert. Val kann jetzt HIGH oder LOW sein. Wenn val HIGH ist, unsere Bildschirme werden beleuchtet. Und wenn val LOW ist, das heißt es gibt keine Bewegung, dann werden Bildschirmlichten ausgeschaltet.

```
void loop()
{
  val = digitalRead(inputPin);  // read input value
  if (val == HIGH)
  {
    lcd.backlight();
    lcd1.backlight();
    if (pirState == LOW)
    {
      // we have just turned on
      Serial.println("Motion detected!");
      // We only want to print on the output change,
     not state
      pirState = HIGH;
    }
  }
  else
  {
    lcd.noBacklight();
    lcd1.noBacklight();
    if (pirState == HIGH)
    {
      // we have just turned of
      Serial.println("Motion ended!");
      // We only want to print on the output change,
     not state
      pirState = LOW;
    }
  }
}
```

V. Real Time Clock

A. *Überblick*

Dies ist eine großartige batteriegepufferte Echtzeituhr (RTC), die es Ihrem Mikrocontroller-Projekt ermöglicht, die

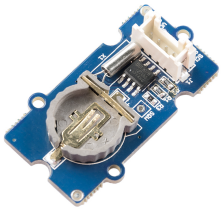

Fig. 13. RTC DS1307

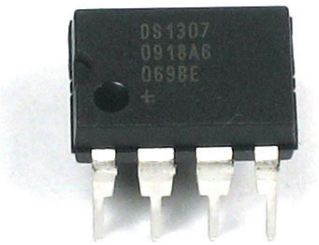

Fig. 14. RTC DS1307

Zeit zu verfolgen, selbst wenn es umprogrammiert wird oder die Stromversorgung ausfällt. Perfekt für die Datenerfassung, die Erstellung von Uhren, Zeitstempel, Timer und Alarme, usw. Der DS1307 ist der beliebteste RTC und funktioniert am besten mit 5V-basierten Chips wie dem Arduino.

- Alle Teile, einschließlich PCB, Header und Batterie sind enthalten
- Schnell zu montieren und zu verwenden
- Kann in jedes Breadboard eingesteckt werden, oder Sie können Drähte verwenden
- Beispielcode und Bibliothek für Arduino mit einer Anleitung auf Dokumentationsseite
- Zwei Befestigungslöcher
- Hält die Zeit für 5 Jahre oder länger

B. Was ist ein RTC

Eine Echtzeituhr ist im Grunde wie eine Uhr - sie läuft mit einer Batterie und behält die Zeit für Sie, auch wenn der Strom ausfällt! Mit einer RTC können Sie lange Zeiträume im Auge behalten, selbst wenn Sie Ihren Mikrocontroller neu programmieren oder ihn vom USB-Anschluss oder einer Steckdose abtrennen.

Die meisten Mikrocontroller, darunter auch der Arduino, haben einen eingebauten Zeitmesser namens millis(), und es gibt auch in den Chip eingebaute Timer, die längere Zeiträume wie Minuten oder Tage verfolgen können. Warum sollte man also einen separaten RTC-Chip haben wollen? Nun, der Hauptgrund ist, dass millis() nur die Zeit seit dem letzten Einschalten des Arduino erfasst. Das bedeutet, dass der Millisekunden-Timer auf 0 zurückgesetzt wird, wenn der Strom eingeschaltet wird. Der Arduino weiß nicht, dass es ”Dienstag” oder ”8. März” ist, er kann nur sagen: ”Es sind 14.000 Millisekunden vergangen, seit ich das letzte Mal eingeschaltet wurde”.

Was wäre, wenn Sie die Zeit auf dem Arduino einstellen wollten? Man müsste das Datum und die Uhrzeit einprogrammieren und könnte die Zeit von diesem Zeitpunkt an zählen lassen. Aber wenn der Strom ausfällt, muss man die Zeit neu einstellen. Ähnlich wie bei sehr billigen Weckern: Jedes Mal, wenn der Strom ausfällt, blinkt 12:00

Während diese Art der grundlegenden Zeitmessung für einige Projekte in Ordnung ist, benötigen einige Projekte wie Datenlogger, Uhren usw. eine konsistente Zeitmessung, die nicht zurückgesetzt wird, wenn die Arduino-Batterie stirbt oder umprogrammiert wird. Daher enthalten wir eine separate RTC! Der RTC-Chip ist ein spezieller Chip, der nur die Zeit misst. Er kann Schaltjahre zählen und weiß, wie viele Tage ein Monat hat, aber er kümmert sich nicht um die Sommerzeit (weil sie sich von Ort zu Ort ändert)

Die RTC, die wir verwenden werden, ist die DS1307. Sie ist preiswert, leicht zu löten und kann jahrelang mit einer sehr kleinen Knopfzelle betrieben werden.

Solange sie mit einer Knopfzelle betrieben wird, wird die RTC lange Zeit fröhlich vor sich hin ticken, auch wenn der Arduino die Stromversorgung verliert oder neu programmiert wird.

C. Verdrahtung

Es gibt nur 5 Pins: 5V GND SCL SDA SQW.

5V wird für die Stromversorgung des RTC-Chips verwendet, wenn Sie die Uhrzeit abfragen wollen. Wenn kein 5V-Signal anliegt, geht der Chip in den Ruhezustand und verwendet die Knopfzelle als Backup. Verbinden Sie GND mit der gemeinsamen Strom-/Datenerde Verbinden Sie den SCL-Pin mit dem SCL-Pin des I2C-Taktgebers an Ihrem Arduino. Bei einem UNO & ’328 basierten Arduino ist dies auch als A5 bekannt, bei einem Mega ist es auch als digital 21 bekannt und bei einem Leonardo/Micro, digital 3 Verbinden Sie den SDA-Pin mit dem I2C-Daten-SDA-Pin Ihres Arduino. Bei einem UNO- und ’328-basierten Arduino ist dies auch als A4 bekannt, bei einem Mega ist es auch als digital 20 und bei einem Leonardo/Micro als digital 2 bekannt. SQW ist der optional.

D. Implementierung im Projekt

Wir benutzen RTC, damit Arduino Zeit rechnet und auf dem Display anzeigt. Startet aber Uhr nicht richtig, dafür haben wir Buttons hinzufügt damit wir Uhrzeit korrekt einstellen können. Buttons diskutieren wir später.

Für RTC braucht man zwei Bibliotheken: einmal DS1307 und ein mal RTClib.h. Hier haben wir diese Bibliotheken importiert.

```
#include "DS1307.h"
#include <RTClib.h>
```

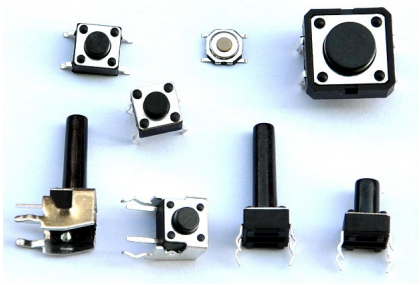

Fig. 15. Buttons

Danach brauchen wir variable für DS1307 Objekt und für Uhrzeitkomponenten wir Stunde, Minute, Jahr, Monat und Tag. Hier sind die Variablen.

```
DS1307 clock; //define a object of DS1307 class

int hourCnt;
int minagg;
int yearCnt;
int meseagg;
int dayagg;
```

Im Setup Funktion starten wir Uhr und setzen wir aktuellste Uhrzeit. Das ist Uhrzeit, in der wir dieses Projekt gemacht haben. Später kann man mit Buttons Uhrzeit ändern.

```
void setup(){
  //Clock
  clock.begin();
  clock.fillByYMD(2022, 1, 22); //Jan 19,2013
  clock.fillByHMS(20, 15, 0); //15:28 30"
  clock.fillDayOfWeek(SAT);//Saturday
  clock.setTime();//write time to the RTC chip
}
```

E. Integration mit Buttons

Push tasten oder Schalter verbinden zwei Punkte in einem Schaltkreis, wenn Sie sie drücken. In diesem Beispiel wird eine LED eingeschaltet, wenn die Taste einmal gedrückt wird, und ausgeschaltet, wenn sie zweimal gedrückt wird.

Wir haben drei Buttons benutzt und alle sind mit input pullup im Setup konfiguriert.

```
  void setup(){
    // Botton
    pinMode(P1,INPUT_PULLUP);
    pinMode(P2,INPUT_PULLUP);
    pinMode(P3,INPUT_PULLUP);
    int menu=0;
  }
```

Eine Taste ist für menu und es gibt insgesamt 6 Menus. Jede ist für set Stunde, Minute, Jahr, Monat, Tag und letzte ist für Speicherung. Hier sieht man, Button 1 input wird durch digitalRead(P1) gelesen und dann Menu entsprechend Funktion ausgeführt.

```
void loop(){
   if(digitalRead(P1) == LOW)
  {
    menu=menu+1;
  }

  if (menu==0)
  {
    DisplayDateTime();

  }
  if (menu==1)
  {
    DisplaySetHour();

  }
  if (menu==2)
  {
    DisplaySetMinute();
  }
  if (menu==3)
  {
    DisplaySetYear();

  }
  if (menu==4)
  {
    DisplaySetMonth();

  }
  if (menu==5)
  {
    DisplaySetDay();

  }
  if (menu==6)
  {
    StoreAgg();
    delay(500);
    menu=0;
  }
  delay(100);
  }

}
```

VI. I2C LCD Display

A. Warum benutzen wir I2C

Wenn Sie jemals versucht haben, ein LCD-Display mit einem Arduino zu verbinden, haben Sie vielleicht bemerkt, dass es eine Menge Pins am Arduino verbraucht. Selbst im 4-Bit-Modus benötigt der Arduino immer noch insgesamt sieben Anschlüsse - das ist die Hälfte der verfügbaren digitalen E/A-Pins des Arduino.

Die Lösung ist die Verwendung eines I2C-LCD-Displays. Es benötigt nur zwei I/O-Pins, die nicht einmal Teil eines Satzes digitaler I/O-Pins sind, und kann auch mit anderen I2C-Geräten gemeinsam genutzt werden.

B. I2C LCD Hardware Übersicht

Ein typisches I2C-LCD-Display besteht aus einem HD44780-basierten Zeichen-LCD-Display und einem I2C-LCD-Adapter. Lernen wir sie nacheinander kennen.

Fig. 16. Pixeln in Display

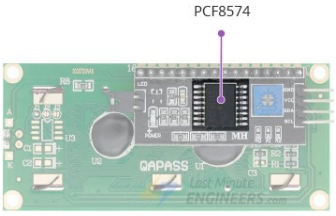

Fig. 17. PCF8574

1) Character LCD Display: Wie ihr Name schon sagt, sind diese LCDs ideal für die Anzeige von Text/Zeichen. Ein 16×2-Zeichen-LCD hat beispielsweise eine LED-Hintergrundbeleuchtung und kann 32 ASCII-Zeichen in zwei Reihen mit je 16 Zeichen anzeigen.

Wenn Sie genau hinsehen, können Sie die kleinen Rechtecke für jedes Zeichen auf dem Display und die Pixel sehen, aus denen ein Zeichen besteht. Jedes dieser Rechtecke ist ein Raster von 5×8 Pixeln.

2) I2C LCD Adapter: Das Herzstück des Adapters ist ein 8-Bit I/O Expander Chip - PCF8574. Dieser Chip wandelt die I2C-Daten von einem Arduino in die vom LCD-Display benötigten parallelen Daten um.

Die Platine ist außerdem mit einem kleinen Trimpot ausgestattet, mit dem der Kontrast des Displays fein eingestellt werden kann.

Darüber hinaus befindet sich auf der Platine ein Jumper, der die Hintergrundbeleuchtung mit Strom versorgt. Um die Intensität der Hintergrundbeleuchtung zu steuern, können Sie den Jumper entfernen und eine externe Spannung an den mit "LED" gekennzeichneten Stift der Stiftleiste anlegen.

3) I2C Address of LCD: Wenn Sie mehrere Geräte auf demselben I2C-Bus verwenden, müssen Sie möglicherweise

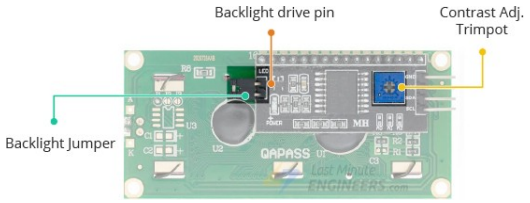

Fig. 18. Konfiguration von Display

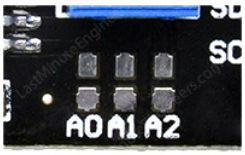

Fig. 19. Adresse setzen für Display

eine andere I2C-Adresse für die Karte einstellen, damit sie nicht mit einem anderen I2C-Gerät in Konflikt gerät.

Dazu verfügt die Platine über drei Lötbrücken (A0, A1 und A2) oder Lötpads.

Jeder dieser Jumper wird zum Festcodieren der Adresse verwendet. Wenn ein Jumper mit einem Klecks Lötzinn kurzgeschlossen wird, wird die Adresse festgelegt.

Ein wichtiger Punkt dabei ist, dass viele Unternehmen den gleichen PCF8574-Chip herstellen: Texas Instruments und NXP Semiconductors, um nur einige zu nennen. Und die I2C-Adresse Ihres LCDs hängt vom Hersteller des Chips ab.

4) Adresse finden im Projekt: Wir haben zwei Displays verwendet und die Adresse zu finden war kompliziert. Wir haben einen kleinen Code geschrieben, damit wir die Adresse finden können. Hier ist der Code:

```
#include <Wire.h>
void setup() {
  Serial.begin (115200);
  while (!Serial)
  {
  }
  Serial.println ();
  Serial.println ("I2C scanner. Scanning ...");
  byte count = 0;
  Wire.begin();
  for (byte i = 8; i < 120; i++)
  {
    Wire.beginTransmission (i);
    if (Wire.endTransmission () == 0)
    {
      Serial.print ("Found address: ");
      Serial.print (i, DEC);
      Serial.print (" (0x");
      Serial.print (i, HEX);
      Serial.println (")");
      count++;
      delay (1); // maybe unneeded?
    } // end of good response
  } // end of for loop
  Serial.println ("Done.");
  Serial.print ("Found ");
  Serial.print (count, DEC);
  Serial.println (" device(s).");
} // end of setup
void loop() {}
```

5) I2C LCD Display Pinout: Ein I2C-LCD hat nur 4 Pins, die die Schnittstelle zur Außenwelt bilden. Die Anschlüsse sind wie folgt: GND ist ein Masse-Pin und sollte mit der Masse des Arduino verbunden werden.

VCC versorgt das Modul und das LCD mit Strom. Schließen Sie ihn an den 5V-Ausgang des Arduino oder an eine separate Stromversorgung an.

SDA ist ein Pin für serielle Daten. Diese Leitung wird sowohl zum Senden als auch zum Empfangen verwendet. Schließen Sie ihn an den SDA-Pin des Arduino an.

SCL ist ein Pin für den seriellen Takt. Dies ist ein Taktsignal, das vom Bus-Master-Gerät geliefert wird. Schließen Sie es an den SCL-Pin des Arduino an.

6) Verdrahtung: Es ist viel einfacher, ein I2C-LCD anzuschließen als ein Standard-LCD. Sie müssen nur 4 Pins anstelle von 12 anschließen. Beginnen Sie mit dem Anschluss des VIN-Pins an den 5V-Ausgang des Arduino und verbinden Sie GND mit Masse.

Nun bleiben noch die Pins, die für die I2C-Kommunikation verwendet werden. Beachten Sie, dass jedes Arduino-Board unterschiedliche I2C-Pins hat, die entsprechend angeschlossen werden sollten. Bei den Arduino-Boards mit dem R3-Layout befinden sich SDA (Datenleitung) und SCL (Taktleitung) auf den Stiftleisten in der Nähe des AREF-Pins. Sie werden auch als A5 (SCL) und A4 (SDA) bezeichnet.

Nachdem Sie das LCD verdrahtet haben, müssen Sie den Kontrast des Displays einstellen. Auf dem I2C-Modul finden Sie ein Potentiometer, das Sie mit einem kleinen Schraubenzieher drehen können.

Stecken Sie den USB-Anschluss des Arduino ein, um das LCD mit Strom zu versorgen. Sie sollten sehen, dass die Hintergrundbeleuchtung aufleuchtet. Drehen Sie nun das Potentiometer, bis die erste Zeile des Rechtecks erscheint.

7) Bibliothek: Um die folgenden Sketche ausführen zu können, müssen Sie eine Bibliothek namens LiquidCrystal_I2C installieren. Diese Bibliothek ist eine verbesserte Version der LiquidCrystal-Bibliothek, die mit Ihrer Arduino-IDE mitgeliefert wird.

Um die Bibliothek zu installieren, navigieren Sie zu Sketch ¿ Include Library ¿ Manage Libraries... Warten Sie, bis der Library Manager den Bibliotheksindex heruntergeladen und die Liste der installierten Bibliotheken aktualisiert hat.

Filtern Sie Ihre Suche durch Eingabe von 'liquidcrystal'. Es sollte ein paar Einträge geben. Suchen Sie nach LiquidCrystal I2C library von Frank de Brabander. Klicken Sie auf diesen Eintrag, und wählen Sie dann Installieren.

C. Imlementierung im Projekt

Der Sketch beginnt mit der Einbindung der LiquidCrystal_I2C-Bibliothek.

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

Anschließend wird ein Objekt der Klasse LiquidCrystal_I2C erstellt. Dieses Objekt verwendet 3 Parameter LiquidCrystal_I2C(address,columns,rows). Hier müssen Sie die Standardadresse in die Adresse ändern, die Sie zuvor gefunden haben, falls sie anders ist, und die Abmessungen des Displays.

```
LiquidCrystal_I2C lcd(0x3F,16,2);
```

Sobald das LiquidCrystal_I2C-Objekt deklariert ist, können Sie auf spezielle Methoden zugreifen, die für das LCD spezifisch sind.

In der Funktion "setup" wird die Funktion init() aufgerufen, um das Objekt lcd zu initialisieren. Anschließend wird die Funktion clear() aufgerufen. Diese Funktion löscht den LCD-Bildschirm und verschiebt den Cursor in die obere linke Ecke. Mit der Funktion backlight() wird die Hintergrundbeleuchtung des LCD-Bildschirms eingeschaltet.

```
lcd.init();
lcd.clear();
lcd.backlight();
```

Als nächstes wird die Cursorposition auf die dritte Spalte und die erste Zeile des LCDs gesetzt, indem die Funktion lcd.setCursor(2,0) aufgerufen wird. Die Cursorposition gibt die Stelle an, an der der neue Text auf dem LCD angezeigt werden soll. Die linke obere Ecke wird als col=0, row=0 betrachtet.

```
lcd.setCursor(2,0);
```

Anschließend wird die Zeichenkette "Hello World!" durch Aufruf der Funktion print() gedruckt.

```
lcd.print("Hello world!");
```

In ähnlicher Weise setzen die nächsten beiden Zeilen den Cursor auf die dritte Spalte und die zweite Zeile und drucken 'LCD Tutorial' auf das LCD.

```
lcd.setCursor(2,1);
lcd.print("LCD Tutorial");
```

Wir haben auch die Adressen gefunden und die Bibliothek eingebunden.

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address 0
    x27, 16 column and 2 rows
LiquidCrystal_I2C lcd1(0x03F, 16, 2); // I2C address
     0x03F, 16 column and 2 rows
```

Im Setup Funktion starten die beide Displays mit Rücklichter.

```
lcd.init(); // initialize the lcd
lcd.backlight();

lcd1.init(); // initialize the lcd 1
lcd1.backlight();
```

Danach haben wir mit clear() und setCurson() Funktionen Informationen von Sensoren auf dem Display Dargestellt.

REFERENCES

[1] https://github.com/adafruit/DHT-sensor-library
[2] https://github.com/semestrinis/Arduino/wiki/DHT22-temperature-humidity-sensor
[3] https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/overview
[4] https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/how-pirs-work
[5] https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/connecting-to-a-pir

[6] https://wiki.seeedstudio.com/Grove-RTC/
[7]