

SmartBierdegg

Achim Herrmann, Achim Däubler

Abstract—Ziel des Projektes SmartBierdegg ist es einen Bierdeckel zu konstruieren, der automatisch mitzählt wieviele Getränke man bereits hat.

Er besteht aus einer Wägezelle, mit der das Gewicht des darauf abgestellten Getränks gemessen wird. Außerdem enthält er einen Mikrocontroller, der die gemessenen Gewichtsdaten erhält und daraus errechnet ob ein neues Getränk abgestellt wurde oder nicht. Wie viele neue Getränke schon abgestellt wurden wird durch LEDs am Bierdeckel veranschaulicht.

I. EINLEITUNG

ZUERST wird der Aufbau des Bierdeckels beschrieben. Es wird erklärt, welche Bestandteile benötigt werden und wie sie funktionieren. Dann zeigen wir unser Platinendesign, das die Bauteile miteinander verbindet. Zum Schluss wird anhand unseres Codes veranschaulicht, wie die Gewichtsmessung funktioniert.

II. VORABÜBERLEGUNGEN

Zunächst wird eine Möglichkeit benötigt, welche die automatische Detektion eines neuen Getränks ermöglicht. Hierfür fiel die Wahl auf Überwachung des aktuellen Gewichts auf dem Bierdeckel. Die Grundlage für die Gewichtsmessung sind Dehnmessstreifen (DMS). Diese ändern bei Dehnung bzw. Stauchung ihren elektrischen Widerstand. Zusammen mit weiteren DMS oder Widerständen kann man damit eine sogenannte Wheatstonesche Messbrücke (Abb. 1) konstruieren. Ersetzt man in Abb. 1 nun einen oder mehrere Widerstände durch DMS, so führen Dehnungen oder Stauchungen auf diesen zu Spannungsänderungen. Befestigt man nun die DMS an einem geeigneten Federkörper so erhält man eine sog. Wägezelle (engl. Load Cell, Abb. 2), welche für das Projekt verwendet wurden. Es ist auch möglich die Messstreifen direkt an dem Bierdeckel zu befestigen, dies wurde allerdings aufgrund der hohen Kosten für einen DMS im Vergleich zu einer Wägezelle verworfen. Um die

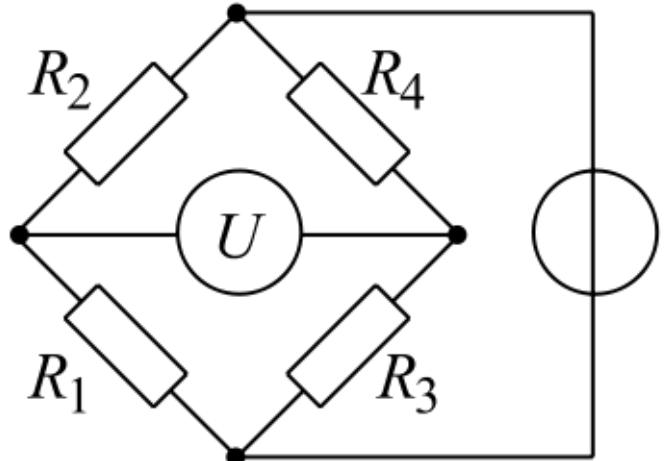


Fig. 1. Wheatstone'sche Brücke

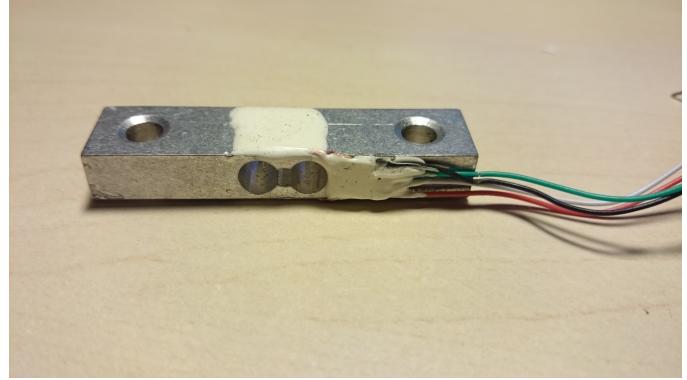


Fig. 2. Wägezelle

Daten der Wägezelle in einem Microcontroller verarbeiten zu können, benötigt man einen Analog-Digital Wandler. Dieser kann bereits in dem Microcontroller vorhanden sein. In diesem Fall wird jedoch noch ein Signalverstärker benötigt. Eine andere Möglichkeit ist die Verwendung eines dedizierten Moduls, wie dem HX711 [1] (Abb. 3). Dieser wandelt das analoge Signal der Wägezelle in einen 24 bit Digitalwert, welcher durch den Microcontroller ausgelesen werden kann.

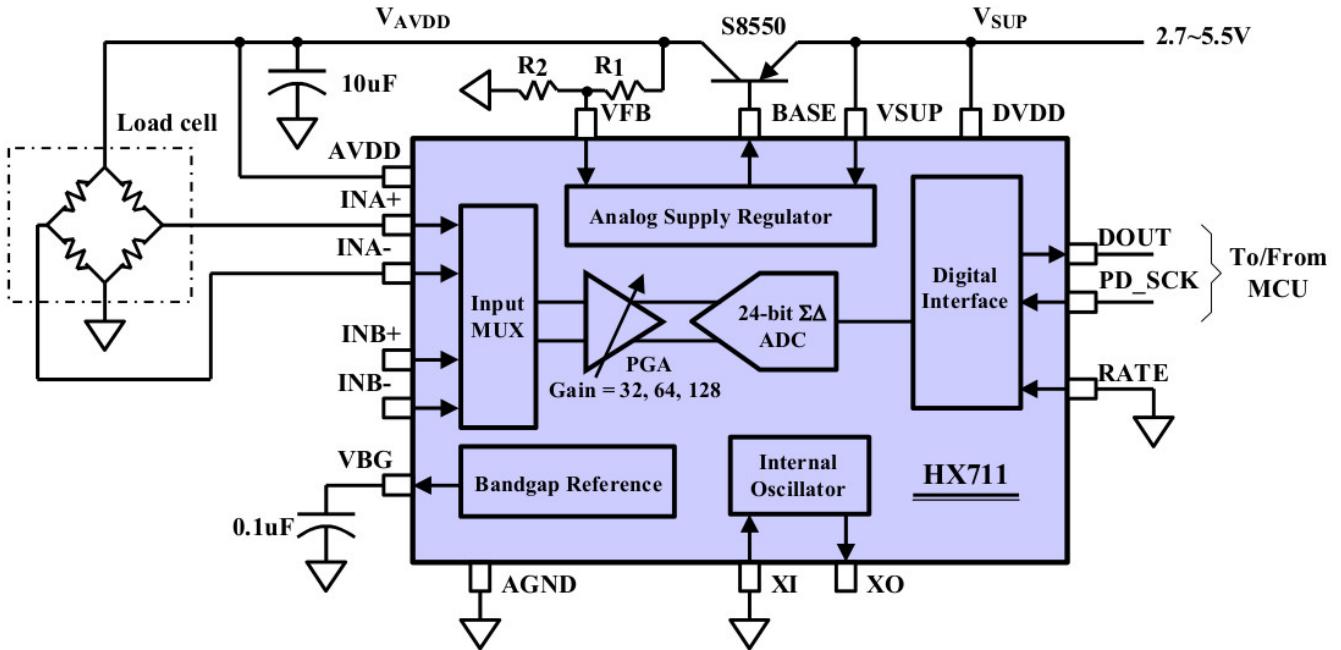


Fig. 4. HX711 Beschaltung

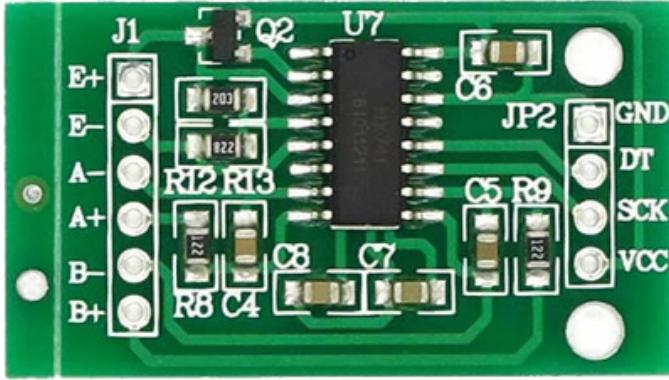


Fig. 3. HX711 Board

III. FUNKTIONSWEISE DES HX711

Der Aufbau der Platine, die den HX711 IC enthält, ist in Fig. 4 dargestellt. Wie man in der Abbildung sehen kann werden die Pins AVDD, AGND, INA+ und INA- und (entspricht E+, E-, A+ und A-) auf der Platine in Abbildung 3) des HX711 mit der Wägezelle verbunden. Jetzt wird gemäß der Wheatstone Brückenschaltung die Spannungsdifferenz zwischen INA+ und INA- gemessen. Dann wird sie verstärkt und vom 24 Bit Analog-zu-Digitalkonverter und in einen Digitalwert umgewandelt. Dieser muss nun ausgelesen werden.

Die Pins PD_SCK und DOUT (entspricht SCK und DT in Fig. 3) werden für den Datenaustausch ver-

wendet. Sobald DOUT von HIGH auf LOW wechselt, sind Daten zur Abholung bereit. Dann müssen 25 positive Taktsignale an PD_SCK geschickt werden (PD_SCK muss mindestens $2\mu s$ lang HIGH und $2\mu s$ LOW sein, also nicht zu schnell lesen!), wobei jedes ein Bit aus DOUT shifted. Begonnen wird dabei mit dem MSB Bit. Der 25. Takt zieht DOUT wieder auf HIGH. Indem weitere Takte gesendet werden (höchstens 27) kann man für das nächste Signal wählen ob Eingang A oder B ausgelesen wird welchen Gain das gelesene Signal bekommen soll.

PD_SCK wird außerdem verwendet um den HX711 resetten. Wenn der Pin länger als $60\mu s$ auf HIGH steht dann wird der Power Down Mode aktiviert. Wenn der Pin wieder auf LOW gezogen wird, dann setzt sich der Chip zurück und geht in den normalen Betriebsmodus über.

IV. AUFBAU

Der SmartBierdeggl besteht aus folgenden Teilen:

- Microcontroller
- Wägezelle
- HX711
- LEDs
- Stromversorgung

Die Wägezelle stammte hierbei von einer Küchenwaage, welche sich auch gut zum Testen eignete. Als Microcontroller wurde anfangs das

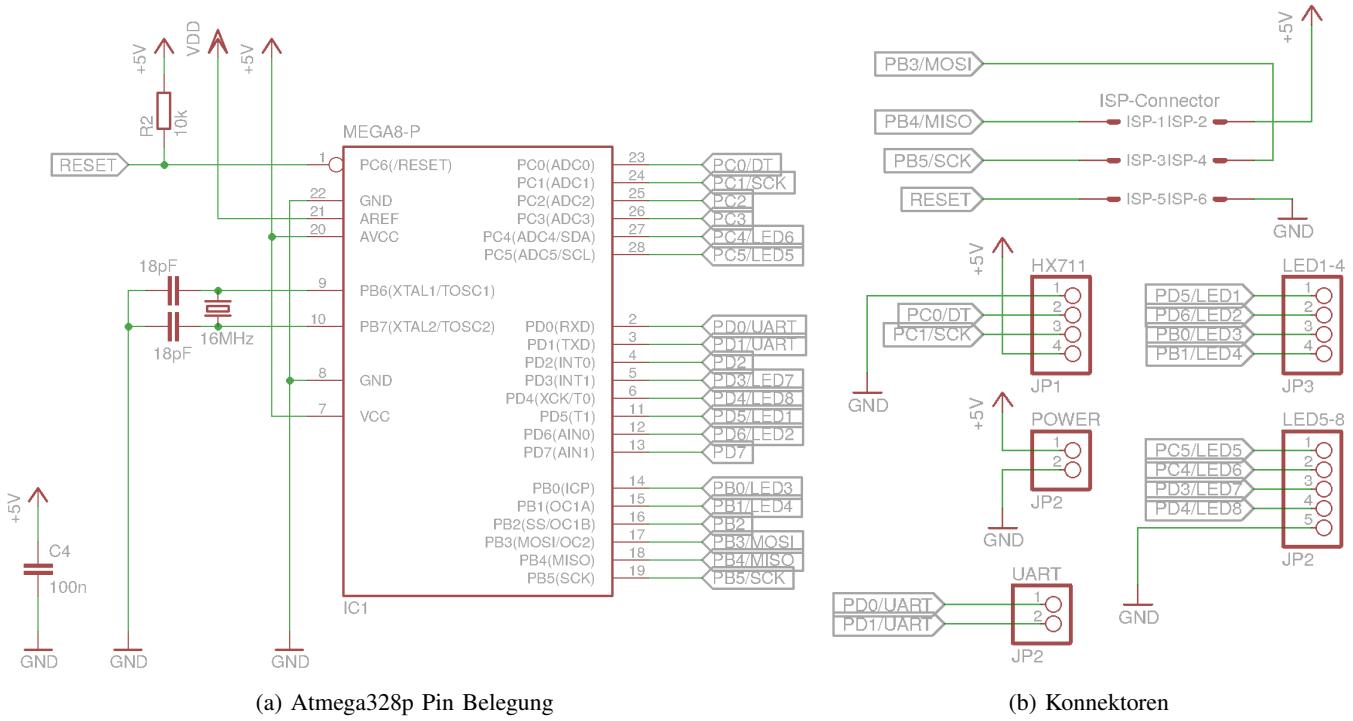


Fig. 5. Beschaltung des Atmega328p

TinyUSBBoard [4] verwendet. Dies wurde für Testzwecke, auf Grundlage des EasyLogger [5], um eine USB-Ausgabe Funktionalität erweitert. Jedoch stellte sich heraus, dass die USB-Ausgabe mit dem Timing des HX711 interferiert. Infolge dessen wurde statt des TinyUSBBoard ein UNO-Board verwendet, welches das Testen über eine serielle Schnittstelle ermöglicht. Sobald das System lief, wurde auf einen separaten Microcontroller umgestiegen. Dies ist aktuell noch ein Atmel ATmega328PA, dieser kann jedoch ohne weiteres durch einen ATmega48 ersetzt werden. Um diesen herum wurde anschließend auf einem Steckboard ein Prototyp für den SmartBierdegg entworfen (Abb. 6). Als Stromversorgung dienen hierbei 3 AA Batterien mit jeweils 1,5V.

V. BESCHALTUNG DES ATMEGA328P

Um uns das Leben zu erleichtern haben wir eine Platine entworfen, die den Atmega328p enthält (siehe Fig. 5a). Die anderen Pins sind mit Konnektoren verbunden (siehe Fig. 5b). Wir haben also auf dem Board, das den Mikrocontroller enthält mehrere Steckplätze für die anderen Komponenten. Das sind also die Steckplätze für:



Fig. 6. Prototyp

- Das Batteriefach, dass so mit der Platine verbunden wird und die Betriebsspannung (VCC) und die Verbindung zu GND für alle Komponenten liefert.
- Den HX711 mit den Pins für VCC, SCK, DT und GND
- Die erste Reihe der LEDs auf dem separaten LED Board
- Die zweite Reihe der LEDs plus einem Pin für GND
- Einen ISP Konnektor
- die Stecker für die UART Schnittstelle

Die letzten beiden tragen nicht zur Funktion unseres Bierdeckels bei aber wurden eingebaut um den Bierdeckel neu programmieren (über den ISP Konnektor über die serielle Schnittstelle) zu können falls man das möchte. Über die UART Schnittstelle kann man das Programm dann debuggen. Das fertige Layout der Platinen kann man in Fig. 9 und Fig. 8 sehen.

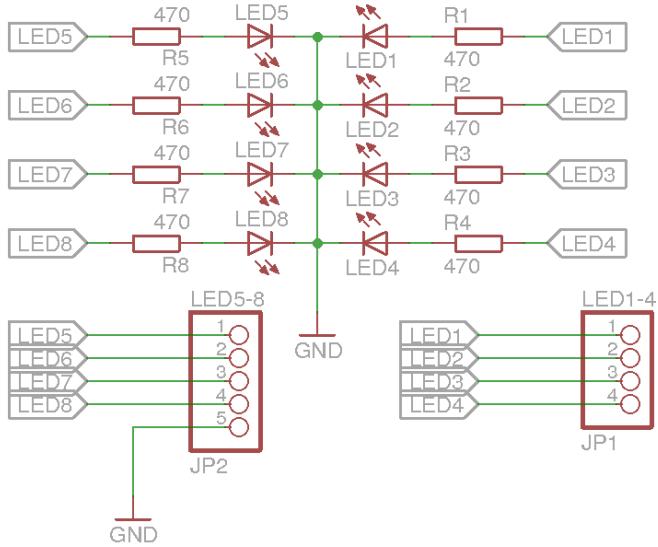


Fig. 7. Schema der LED Platine

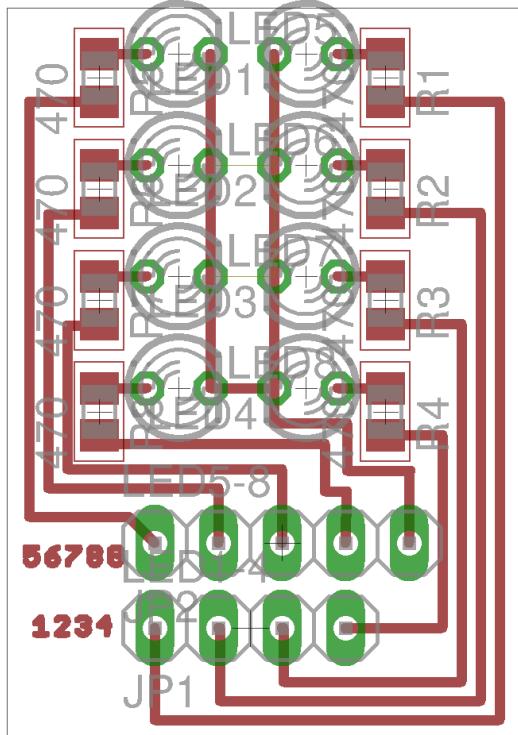


Fig. 8. Platine mit den Zähler LEDs

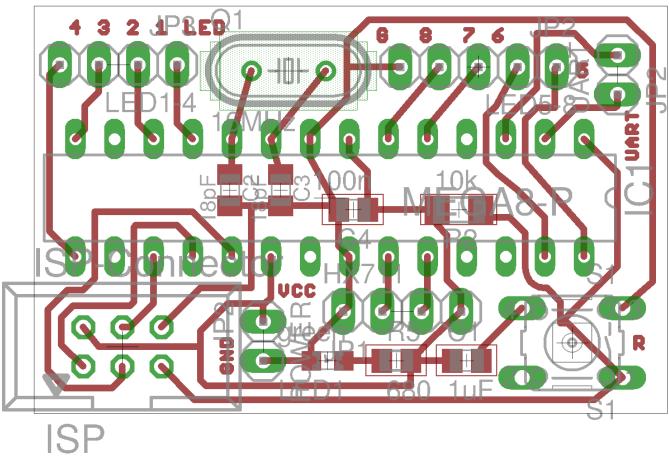


Fig. 9. Platine mit Atmega328p

VI. μ C PROGRAMM

Zur Programmierung des Microcontrollers wurde nach Abbildung 10 das UNO-Board als ISP genommen. Hierbei ist jedoch darauf zu achten, RST und GND durch einen zusätzlichen $100\mu\text{F}$ Kondensator zu verbinden, um zu verhindern, dass sich der Uno selbst neu programmiert.

In folgendem Code wird unser HX711 initialisiert indem wir ihn einmal resetten. Wir setzen also den Pin PD_SCK (SCLK im Code) $100\mu\text{s}$ lang auf HIGH. Dann lesen wir 32 mal das Gewicht aus (hx711_getValue() wird 32 mal aufgerufen) und nehmen den gemittelten Wert als Offset (entspricht dem Gewicht des leeren Glases, das zu dem Zeitpunkt auf dem Bierdeckel stehen muss).

```

1 void hx711_init(void)
{
3   DDR_HX |= SCLK;
4   DDR_HX &= ~SDI;
5
6   // reset HX711
7   PORT_HX |= SCLK;
8   _delay_us(100);
9   PORT_HX &= ~SCLK;
10
11  hx711_averageValue(32);
12  // calibration
13  // _offset is weight of empty glass
14  offset = hx711_averageValue(32);
15  _scale = 371;
}

```

In hx711_getValue() wird jetzt gewartet bis DOUT (=SDI im Code) auf LOW wechselt. Dann werden 24 Taktsignale an PD_SCK (=SCLK im

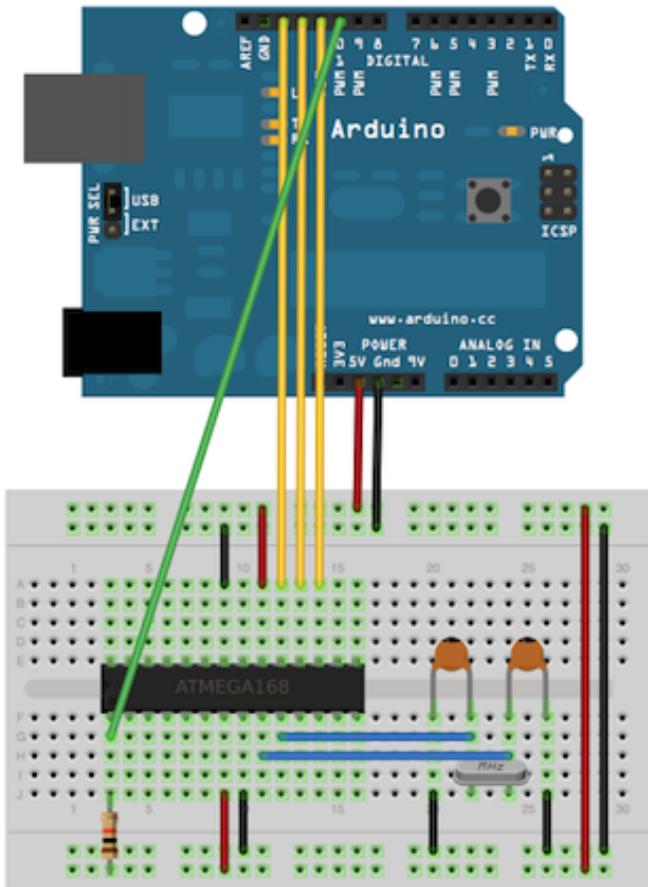


Fig. 10. UNO als ISP

Code) geschickt um die Daten Bit für Bit auszulesen. Mit dem 25. Takt wird DOUT wieder auf HIGH gesetzt. Da kein weiterer Takt gesendet wurde, verwenden wir den Input Kanal A des HX711 mit einem Gain von 128 für das nächste Mal.

```

1 uint32_t hx711_getValue()
2 {
3     uint8_t i = 0;
4     uint32_t data = 0;
5
6     // When output data isn't ready
7     // for retrieval, digital output pin
8     // DOUT (=SDI) is high
9     while (PIN_HX & SDI);
10
11    // read data bit by bit by sending 24
12    // positive clock pulses to PD_SCK pin
13    for (i = 24; i--;) {
14        PORT_HX |= SCLK;
15        data |=
16            (uint32_t)((PIN_C & SDI) ? 1 : 0) << i;
17        PORT_HX &= ~SCLK;
18    }
19 }
```

```

21 // 25th pulse sets DOUT (=SDI)
22 // pin back to HIGH
23 PORT_HX |= SCLK;
24 PORT_HX &= ~SCLK;
25
26 // no other pulses are sent
27 // so we will be using input channel A
28 // with a gain of 128
29
30 data ^= 0x800000;
31
32 return data;
33 }
```

Die Gewichtsdaten liegen jetzt als 24Bit Wert vor. Dieser muss in Gramm umgerechnet werden. Das geschieht mit einem empirisch bestimmten Skalierungsfaktor.

```

1 int main(void)
2 {
3     init();
4     uint8_t n = 0;
5     bool empty = false;
6     // blink once to indicate
7     // initialization has finished
8     led_show(25); timer_wait(1); led_show(0);
9     for(;;){
10         int32_t valg;
11         if (isEmpty((valg = hx711_get_mg()) / 1000)){
12             // prevent false empty indication
13             // while drinking
14             sleep16m(1);
15             // blink while glass is empty
16             valg = hx711_get_mg() / 1000;
17             while (isEmpty(valg)){
18                 empty = true;
19                 led_blink(n);
20             }
21         }
22         if (empty && isFull(valg)){
23             ++n;
24             empty = false;
25         }
26         led_show(n);
27         sleep(1);
28     }
29 }
```

In der main() wird nun im Sekundentakt geprüft, ob das abgestellte Glas leer oder voll ist. hx77_get_mg() ruft dabei hx711_getValue() 32 mal auf um einen gemittelten Wert zu erreichen und Schwankungen so herauszufiltern. Solange das Glas leer ist blinken die LEDs. Als leer wird das Glas erkannt, wenn der Inhalt mit einem Gewicht zwischen -50g und 50g gemessen wird. Als voll wird

es erkannt wenn der Inhalt über 480g wiegt. Wird nun erkannt, dass vom leeren in den vollen Zustand gewechselt wurde, wird daraus geschlossen, dass ein neues Bier da ist. Dann wird die LED Anzeige eins hochgezählt. Die LEDs werden einzeln über Pins am Atmega328p angesteuert. Anschließend geht der Microcontroller für eine Sekunde in den Stromsparmodus.

VII. VERWENDUNG

Sobald man den Bierdeckel anschaltet, muss das leere Glas auf dem Bierdeckel stehen. Der HX711 wird dann durch den Atmega initialisiert und es wird das Gewicht des leeren Glases als Offset errechnet, damit man das Gewicht des Glases nicht mitzählt. Wird nun ein volles Glas abgestellt, wird erkannt dass vom leeren Zustand in den vollen gewechselt wurde, also ein neues Bier da ist. Dann wird die LED Anzeige eins hochgezählt. Die Anzeige funktioniert, wie beim klassischen Bierdeckel, mit den Strich/Zaun Zählsystem. Eine 4er Reihe der LEDs entspricht also den Strichen und eine den Zäunen. Wenn 4 Strich-LEDs leuchten und eins hochgezählt wird, dann gehen diese aus und eine weitere Zaun-LED fängt an zu leuchten.

VIII. AUSBLICK

Zum aktuellen Stand sind die Einzelteile lose miteinander verbunden. Deswegen ist ein Case geplant, in dem wir unsere Bestandteile unterbringen wollen. Dieses sollte möglichst klein sein auch wenn es schwer werden dürfte den Durchmesser und die Höhe eines Standard Bierdeckels einzuhalten. Auch wäre es wünschenswert von dem ATMega auf einen günstigeren ATTiny umzusteigen, wobei man einen Möglichkeit benötigt mehrerer LEDs mit weniger Pins anzusteuern. Denkbare Erweiterungen wären ein Funk Modul (z.B. Bluetooth) einzubauen, über welches der Bedienung signalisiert werden kann, dass das Glas leer ist. Außerdem wäre es dadurch denkbar eine App zu erstellen, welche es ermöglicht die Kosten anzuzeigen. Oder auch, nach Eingabe von Körperdaten, zu berechnen ob und wann man wieder fahrtauglich ist. Eine weitere Erweiterung wäre es Getränke zu unterscheiden, entweder durch direkt Eingabe in der App, oder durch eine Codierung bzw. RFID Chip am Glas.

REFERENCES

- [1] AVIA Semiconductor, "24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales", HX711 Datasheet
- [2] Atmel, "ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KBYTES", Datasheet, Nov. 2015
- [3] https://de.wikipedia.org/wiki/Wheatstonesche_Messbrücke
- [4] <https://github.com/i7sid/diy-tinyusbboard>
- [5] <https://www.obdev.at/products/vusb/easylogger.html>