

Abstandsmessung mit dem SparkFun ToF Range Finder Sensor - VL6180

Gruppennummer 3



Informationstechnik Labor Sommersemester 2016

Prof. J. Walter

Gruppenmitglieder:
Bramiantomo Hosea
Boniface Mwangi

42968
39225

Inhalt

Inhalt	1
1. Einleitung	2
1.1. Problemstellung	2
1.2. Stand der Technik	2
1.3. Aufgabenstellung	2
2. Anforderungsliste	3
3. Blackbox	4
4. Blockschaltbild	5
5. Zeitplan	6
6. Stückliste	7
7. Aufbau	9
8. Erste Inbetriebnahme von NodeMCU mit Arduino IDE	10
9. Programmierung, Debugging und die Kalibrierung	13
10. Funktionsprinzip	14
11. Abbildungsverzeichniss	16
12. Tabellerverzeichnis	17
13. Quellenverzeichnis	18
14. Anhang	19

1. Einleitung

1.1. Problemstellung

Es gibt zahlreiche Sensoren, die in unterschiedlichen Arten die Abstandsmessung ermöglichen. Dabei spielt die Schnelligkeit, Genauigkeit, präzise Positionierung und Detektion verschiedener Materialien und Beständigkeit gegen Störungen(Rauschen) eine sehr wichtige Rolle bei der Auswahl von geeigneten Sensoren.

Darüber hinaus gibt es auch verschiedene Anzeige Methoden. Der Nutzer möchte je nach Anwendungsbereich die leichteste und bequemste Methode zur Verfügung haben. Jedoch gibt wenige gut optimierte Produkte die das Ganze als ein Paket sich anbieten lassen.

1.2. Stand der Technik

Es gibt schon einige technischen Lösungen die mehr oder weniger Daten von Sensoren anzeigen und gleichzeitig abspeichert. Die meisten davon sind DIY-Projekte. Für die Datenübertragung zwischen Sensor und Micro-computer/Controller wird das standardisierte I²C Protokoll verwendet. Hierfür ist entsprechender Treiber sowohl auf dem Sensor als auch am Mikrocontroller notwendig.

Welche IDE/SDK für die Programmierung zum Einsatz kommt, hängt stark von der Sprache mit der die Firmware geschrieben wurde ab. Die meisten bevorzugen Arduino IDE, Eclipse und Intel XDK. Besonders bei der Intel XDK bietet sich die Möglichkeit für Cross-Plattform App-Entwicklung.

Beispiele von App sind Trebla(Wi-Fi) und getBlue(Bluetooth). Die Lösungen unterscheiden sich von der Art von Sensoren, Übertragungsart(Bluetooth, Wifi,...) und App-Plattform(Android, Windows, iOS,...).

Manche Platine haben schon integrierte WLAN Module wie zum Beispiel das ESP8266 und dadurch sehr praktisch für die Drahtlose Kommunikation.

1.3. Aufgabenstellung

Im Rahmen des Informationstechnik Labor SS206 soll eine kostengünstige Lösung entwickelt bei der ein Abstand zwischen zwei Objekten mit Hilfe des Sparkfun ToF Range Finder VL6180s gemessen und gleichzeitig auf einem Smartphone angezeigt wird. Zuerst soll eine Recherche durchgeführt werden um der aktueller Trend herauszufinden.

Zwischen dem Sensor und Smartphone sitzt NodeMCU 1.0 (ESP8266-12E). Die Entwicklungsumgebung(IDE/SDK) ist frei wählbar. Jedoch ist für die App-Entwicklung ist Intel XDK einzusetzen.

Die Datenübertragung vom Sensor zum NodeMCU 1.0 (ESP8266-12E) erfolgt durch I²C. Dabei ist es wichtig, herauszufinden, ob das Protokoll Software- oder Hardwaremäßig an den jeweiligen Bauteil zu implementieren ist. Des Weiteren soll eine App-Entwickelt werden die der momentan gemessener Werte anzeigt.

2. Anforderungsliste

Informationstechnik Labor Fakultät MMT Hochschule Karlsruhe Technik und Wirtschaft	Anforderungsliste für Abstandsmessung mit dem SparkFun ToF Range Finder Sensor - VL6180	Auftrag: MTB732 Sommersemester 2016
Art	Anforderung	Wert/Daten
Datenübertragung		
J/N	Datenübertragung zwischen Sensor und Mikrokontroller	
J/N	Drahtlose Datenübertragung zwischen Mikrokontroller und Smartphone	
Abstandsmessung		
J/N	Bestimmung vom Abstand	
J/N	Validation der Abstandsmessung	
Elektronik		
J/N	Modulare Bauweise: Trennung von Datenverarbeitung und Sensorik	
W	Verwendung des SparkFun ToF Range Finder Sensor - VL6180	
W	Verwendung des NodeMCU 2.0 (ESP8266-E2)	
W	Spannungsversorgung über Akku	Max. 5.0 V
W	Geringer Energieverbrauch bzw. hohe Akkulaufzeit	20 -50 Betriebsstunden
Mechanik		
W	Gehäuse aus Kunststoff	
F	Maximale Abmessung des Aufbaus	10 cm X 10 cm
Software		
W	Programmierung in Javascript/Html5 für die Anzeige auf Smartphone	
W	I2C Implementierung mit C++	
Restriktionen		
F	Entwicklungsende	6.5.2016
Anforderungsarten: J/N – Unbedingt; F – Tolerierte Forderungen; W- Wunsch		

3. Blackbox

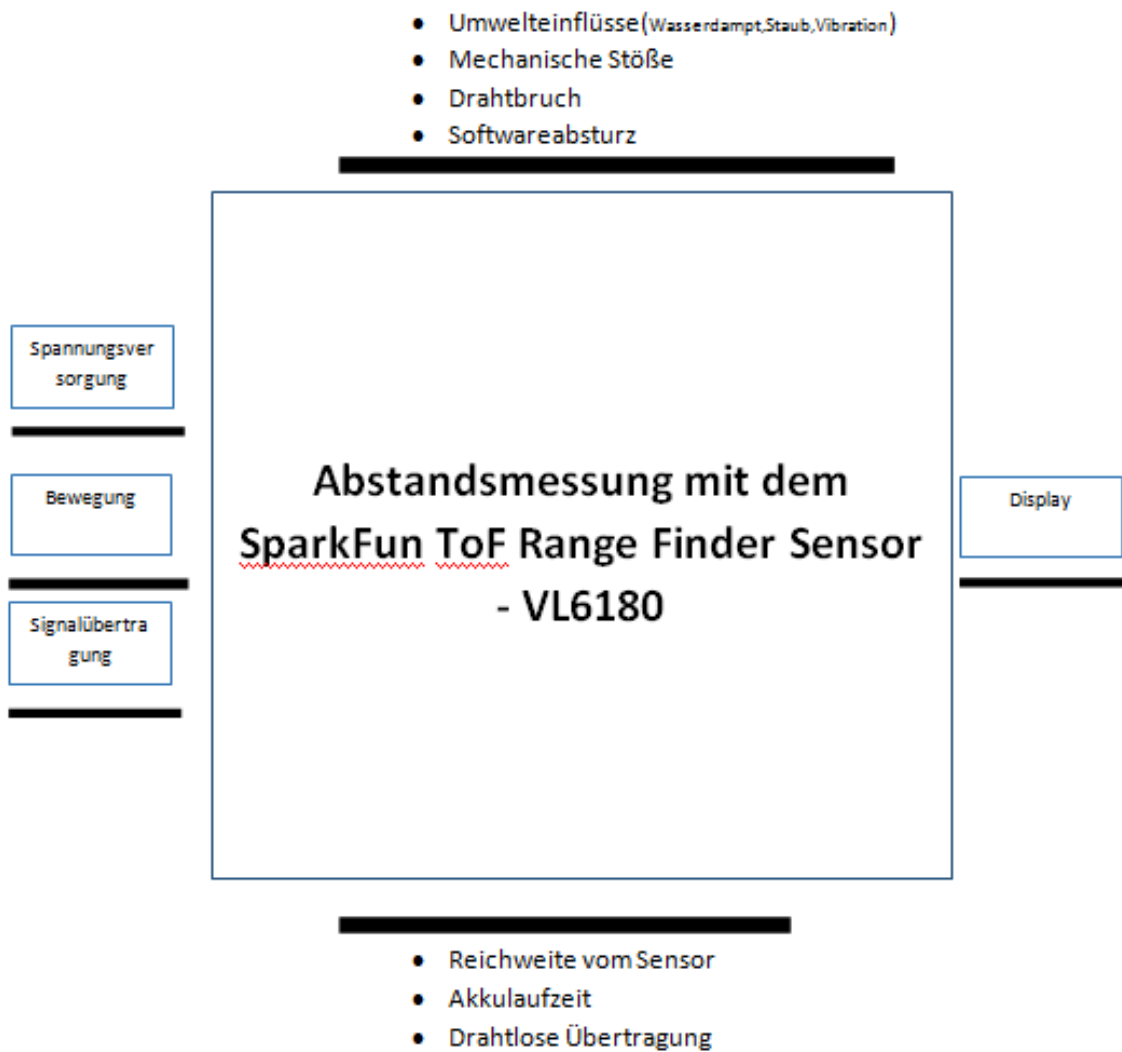


Abbildung 1 Black Box

4. Blockschaltbild

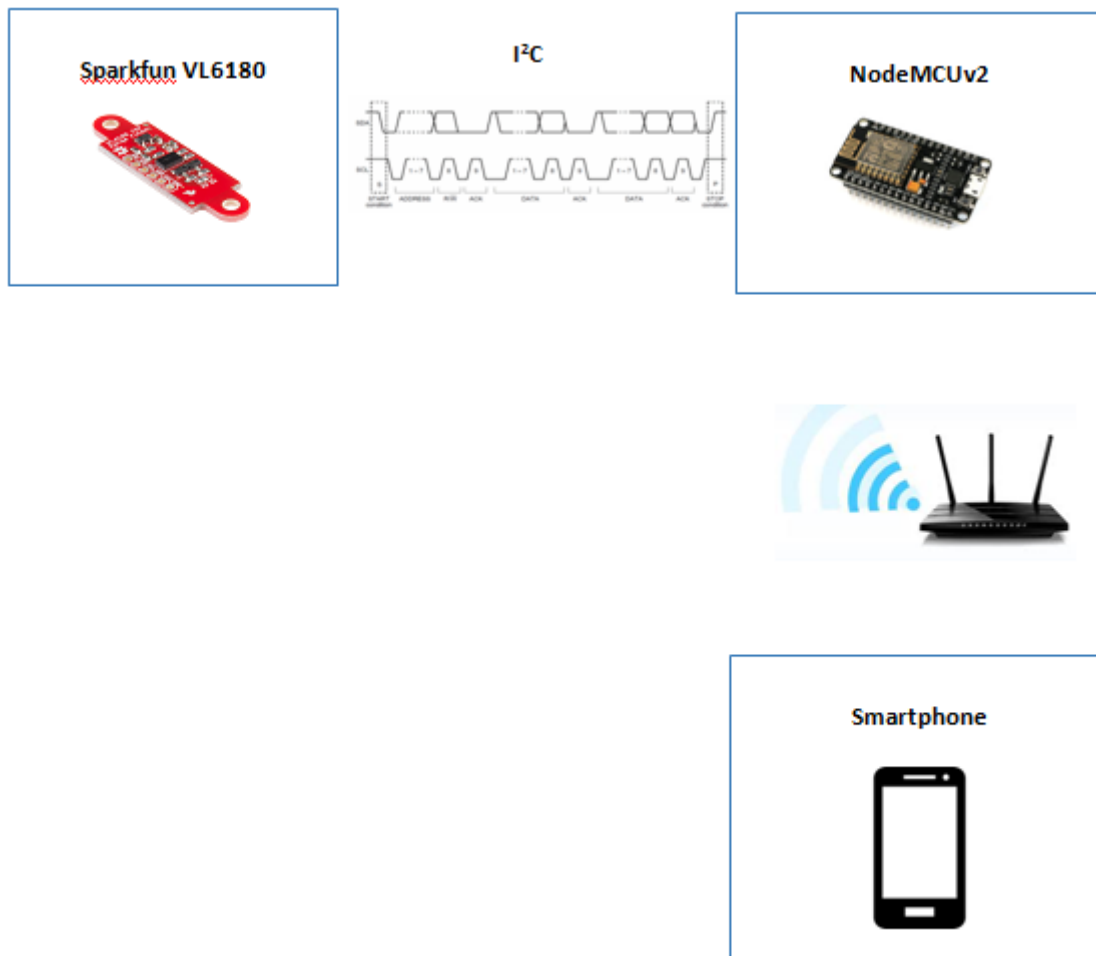


Abbildung 2 Blockschaltbild



5. Zeitplan

Tabelle 1 : Zeitplan

Informationstechnik Labor Fakultät MMT Hochschule Karlsruhe Technik und Wirtschaft		Zeitplan für Abstandsmessung mit dem SparkFun ToF Range Finder Sensor – VL6180								MTB732 Sommersemester 2016 18.3.2016	
Nr.	Aktivitäten	2016									Jahr
		März			April				Mai	Monat	
		11	12	13	14	15	16	17	18	Kalenderwoche	
1	Projektdefinition erstellen	X									
2	Produktrecherche	X	X								
3	Bestellung der Bauteile		X								▽ 1
4	Inbetriebnahme		X	X							
5	Konfiguration - Signalübetragung			X	X	X					
6	App-Entwicklung				X	X	X				▽ 2
7	Testphase						X	X	X		
8	Abschlusspräsentation								X		▽ P
		Meilensteine : ▽ 1 ▽ 2									Präsentation: ▽ P

6. Stückliste

Tabelle 2: Stückliste

Bauteilbezeichnung	Erläuterung
NodeMCU	 <p><i>Abbildung 3 NodeMCU ESP8266-12E</i></p> <p>Hersteller: ESP8266 Opensource Community</p> <p>Eigenschaften:</p> <ul style="list-style-type: none"> • Bemaßung: 49 x 24.5 x 13mm • 5v Vcc(MicroUSB/Akku) • Wifi-Module mit onboard Antennae • 15-Pins(GPIOs, SPI, ADC, UART, GND, Power) • USB-UART Schnittstelle/Treiber • NCP1117 3.3VDC Pegelwandler • 4MB SPI Flashspeicher • Reset und Flash Jumper
SparkFun ToF Range Finder Sensor - VL6180	 <p><i>Abbildung 4 SparkFun ToF Range Finder Sensor - VL6180</i></p> <p>Hersteller: Sparkfun Electronics</p> <p>Eigenschaften:</p> <ul style="list-style-type: none"> • 3-in-1 Module <ul style="list-style-type: none"> – IR Emitter – Range Sensor – Ambient Light Sensor • Messbereich bis zu 10cm • I²C Interface • Two Programmable GPIO • Sharp Sensor Board Layout

Breadboard und Drähte

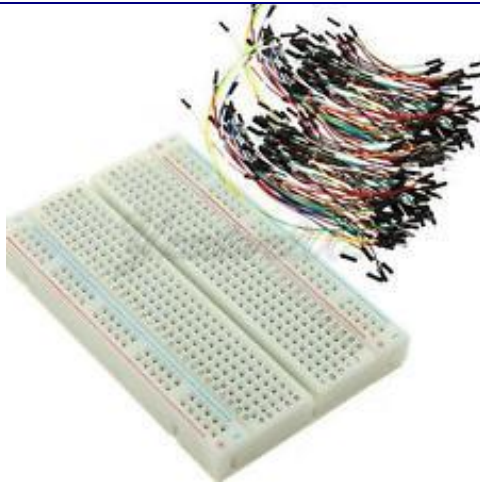


Abbildung 5 Breadboard und Drähte

Micro-USB Kabel



Abbildung 6 Micro-USB Kabel

7. Aufbau

Die Abbildung zeigt den Prinzipiellen Aufbau des Projekts. Spezielle für I²C werden die Pins D1(SCL) und D2(SDA) auf dem NodeMCU mit SCL und SDA auf dem Sensor verbunden. Die Drähte(Orange – SCL, Rot - SDA) sind kurz um Signalstörung zu vermeiden. Der Sensor hängt seitlich damit man das Breadboard beim Messen hin und her schieben kann. Die rote(3.3v) und blaue Drähte dienen zur Versorgung vom Sensor. Allerdings hat der Sensor einen Pegelwandler der die Spannung auf die Sollspannung von 2.8V reduziert.

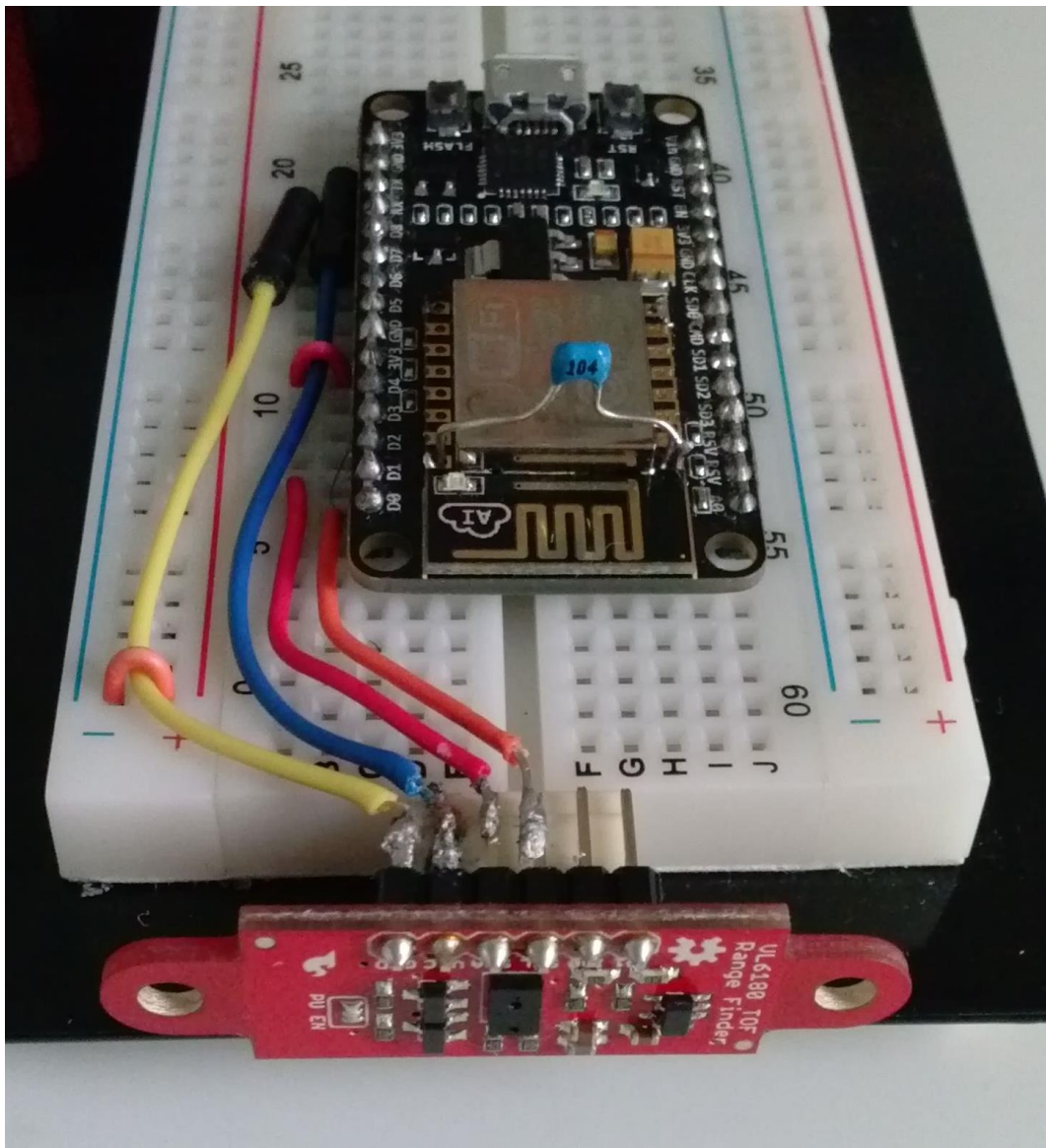


Abbildung 7: Aufbau

8. Erste Inbetriebnahme von NodeMCU mit Arduino IDE

- I. Arduino IDE herunterladen(<https://www.arduino.cc/en/Main/Software>) und installieren.
- II. NodeMCU anstecken.
- III. Arduino IDE starten.
- IV. Unter Files -> Preferences, der Link(http://arduino.esp8266.com/stable/package_esp8266com_index.json) in den Eingabefeld(Additional Boards Manager URLs) eingeben und auf ok klicken.

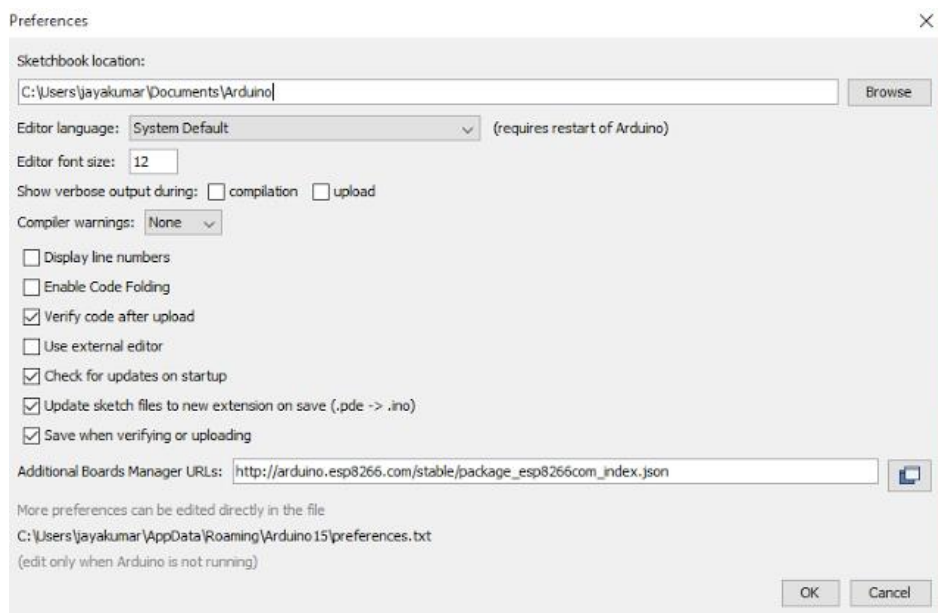


Abbildung 8 Preferences Einstellung

- V. Unter Tools -> Board: " " -> Boards Manager... , esp8266 by esp6266 community auswählen und installieren.

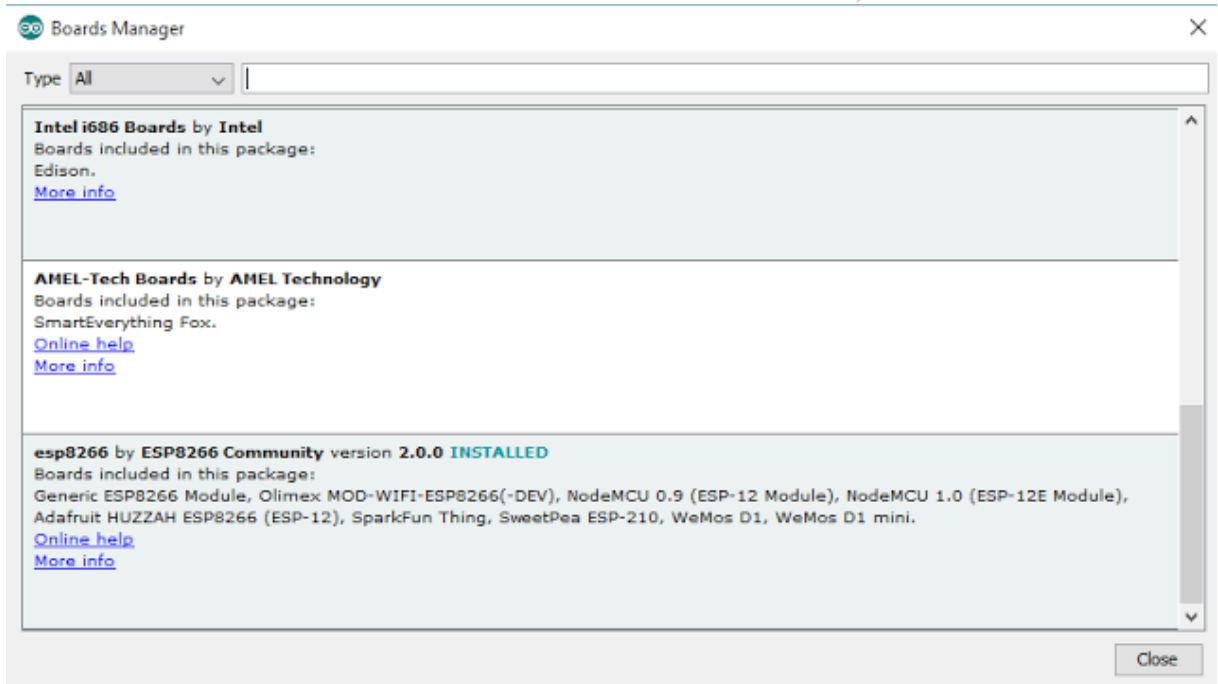


Abbildung 9 Boards Manager

VI. Unter Tools -> Board, nach dem NodeMCU 1.0(ESP-12E Module) suchen und auswählen.

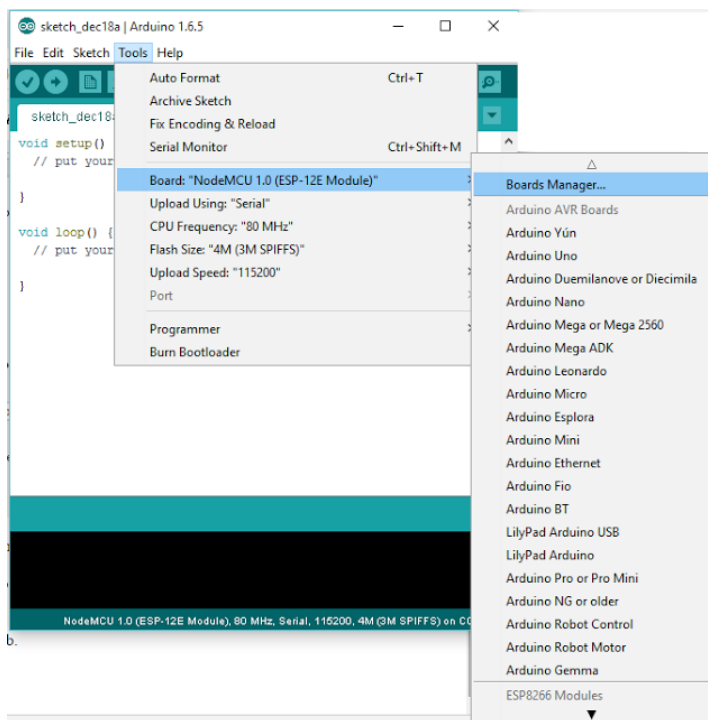


Abbildung 10 Board auswählen

VII. Damit ist die Inbetriebnahme abgeschlossen.

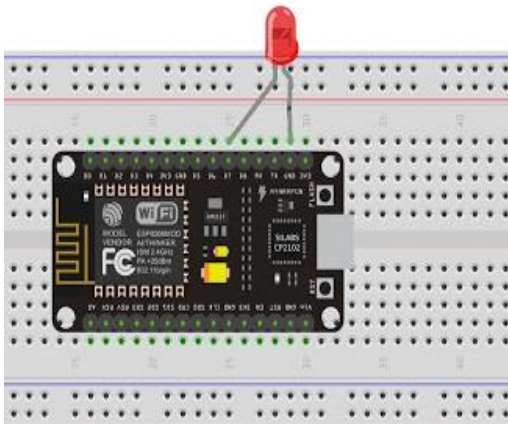


Abbildung 11 Einstellung des Boards

```
1 void setup() {  
2   // initialize digital pin 13 as an output.  
3   pinMode(13, OUTPUT);  
4 }  
5  
6 // the loop function runs over and over again forever  
7 void loop() {  
8   digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)  
9   delay(1000);           // wait for a second  
10  digitalWrite(13, LOW);  // turn the LED off by making the voltage LOW  
11  delay(1000);           // wait for a second  
12 }
```

Abbildung 12 LED-Steuerung Code

Um festzustellen dass die Inbetriebnahme erfolgreich war, haben wir eine Diode folglich steuern lassen. Damit wird die Diode mit einer Takt von einer Sekunde ein und aus geschaltet.

9. Programmierung, Debugging und die Kalibrierung

Nach dem Aufbau ging es weiter in Richtung Softwarebereich. Der NodeMCU unterstützt neben die native Firmware (in Lua programmiert), auch andere Firmware unter anderem ESP8266 (Javascript) und ESP8266 Arduino (C++). Wir haben ESP8266 Arduino Firmware verwendet da der Sparkfun ToF Range Finder Sensor VL6180 eine Bibliothek (C++) hat. Die Bibliothek ist Allerdings für Arduino Uno optimiert und hat leider nicht mit NodeMCU funktioniert. Aus diesem Grund blieb nichts anderes als eigener Code zu schreiben der vor allem I²C Protokoll verwirklichen sollte. Dabei sollte auch ein Webserver im dem Module ESP8266 Konfiguriert sein um die Sensor Daten über Wi-Fi übertragen zu können.

Beim Testen vom Code gab eine Fehlermeldung (Watchdog Reset) die wir mit einem Kondensator (0.1µF) beseitigen könnten. Auf dem Abbildung 7: Aufbau sieht man den blauen Kondensator der am Vcc und GND vom ESP8266 Module sitzt. Diese Fehlermeldung erscheint wenn der NodeMCU in den Restart-Modus wechselt. Wir gehen davon aus das die Spannung-Stabilisierung an der Versorgung vom ESP8266 schwach ist.

Der VL6180 hat beim ersten Blick nicht den vom Sparkfun genannten Wert von 200 mm erreicht. Der Hersteller STI Technologies laut Datenblatt erwähnt jedoch 10 cm. Wir haben beim ersten Funktionstest den Wert 70 mm erreicht. Um den erwünschten Wert von 100 mm zu erreichen müssten wir die Kalibrierung wie folgt durchführen.

2.12.3 Offset calibration procedure

Complete steps 1-3 to see if part-to-part offset calibration is required.

1. Position a white target (88% reflectance^(g)) at a distance of 50 mm from the top of the cover glass.
2. Perform a minimum of 10 range measurements and compute the average range (from `RESULT__RANGE_VAL{0x62}`).
3. If the average range is within the 50 ± 3 mm, offset calibration is not required. Otherwise, complete this calibration procedure.
4. Set `SYSRANGE__PART_TO_PART_RANGE_OFFSET{0x24} = 0`.
5. Perform a minimum of 10 range measurements and compute the average range (from `RESULT__RANGE_VAL{0x62}`).
6. Calculate the part-to-part offset as follows:

$$\text{part-to-part offset} = 50 \text{ mm} - \text{average range}$$

7. Write the part-to-part offset result (in two's complement notation) to `SYSRANGE__PART_TO_PART_RANGE_OFFSET`.

Abbildung 13 Kalibrierung



Abbildung 16 App Layout

Die App hat einen Eingabebereich und zwei Knöpfe (Start und Exit). In dem Eingabebereich soll man URL mit dem Format `http://IP-Adresse/range` eingeben. Mit dem Start-Knopf kann man die Messwerte anzuzeigen und mit dem Exit-Knopf die App schliessen. Wenn URL richtig eingegeben und Start-Knopf gedrückt wird, dann werden die Messdaten in folgende Fenster angezeigt:



Abbildung 17 Angezeigte Messdaten

Solange Zurück-Knopf nicht gedrückt wird, werden Messdaten jede 3 Sekunde aktualisiert und angezeigt. Sobald der Knopf gedrückt wird, schliesst die App dieses Fenster und springt zurück zu Anfangsfenster.

11. Abbildungsverzeichnis

Abbildung 1 Black Box	4
Abbildung 2 Blockschaltbild	5
Abbildung 3 NodeMCU ESP8266-12E.....	7
Abbildung 4 SparkFun ToF Range Finder Sensor - VL6180.....	7
Abbildung 5 Breadboard und Drähte	8
Abbildung 6 Micro-USB Kabel	8
Abbildung 7: Aufbau.....	9
Abbildung 8 Preferences Einstellung.....	10
Abbildung 9 Boards Manager	11
Abbildung 10 Board auswählen.....	11
Abbildung 11 Einstellung des Boards	12
Abbildung 12 LED-Steuerung Code	12
Abbildung 13 Kalibrierung.....	13
Abbildung 14 Ssid und Password.....	14
Abbildung 15 Serial Monitor	14
Abbildung 16 App Layout	15
Abbildung 17 Angezeigte Messdaten.....	15

12. Tabellerverzeichnis

Tabelle 1: Zeitplan	6
Tabelle 2: Stückliste	7

13. Quellenverzeichnis

- [1] <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/DM00112632.pdf>, [21.3.2016, 10:00]
- [2] https://github.com/sparkfun/ToF_Range_Finder_Sensor-VL6180/tree/V_H1.0_L1.1.0 [21.3.2016, 10:30]
- [3] https://github.com/sparkfun/SparkFun_ToF_Range_Finder-VL6180_Arduino_Library/tree/V_1.1.0 [21.3.2016, 11:00]
- [4] https://learn.sparkfun.com/tutorials/vl6180-hookup-guide?_ga=1.184245604.317177452.1462455835 [21.3.2016, 11:20]
- [5] https://cdn.sparkfun.com/datasheets/Sensors/Proximity/VL6180_Sensor_v10.zip [21.3.2016, 11:50]
- [6] <https://learn.sparkfun.com/tutorials/light> [21.3.2016, 12:15]
- [7] <https://learn.sparkfun.com/tutorials/i2c> [21.3.2016, 13:00]
- [8] <http://esp8266.github.io/Arduino/versions/2.0.0/doc/installing.html> [24.3.2016, 14:00]
- [9] <http://www.seeedstudio.com/recipe/1107-getting-started-with-nodemcu-devkit-in-arduino-ide.html> [24.3.2016, 15:00]
- [10] <http://learn.acrobotic.com/tutorials/post/esp8266-getting-started#step-4-lnk> [24.3.2016, 11:00]
- [11] <http://www.espruino.com/EspruinoESP8266> [20.3.2016, 09:00]
- [12] <http://www.esp8266.com/> [20.3.2016, 10:00]

14. Anhang

I. Code in Arduino IDE

```
/*
 * Author: Informationstechnik Labor Gruppe 3, SS2016
 * Version: 1.0
 * Release: 29.4.2016
 * Partly based on Sparkfun https://github.com/sparkfun/SparkFun\_ToF\_Range\_Finder-VL6180\_Arduino\_Library
 */

#include <Wire.h>
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

#define VL6180xx_FAILURE_RESET -1

#define VL6180xx_SYSTEM_MODE_GPIO0      0x010
#define VL6180xx_SYSTEM_MODE_GPIO1      0x011
#define VL6180xx_SYSTEM_HISTORY_CTRL    0x012
#define VL6180xx_SYSTEM_INTERRUPT_CONFIG_GPIO 0x014
#define VL6180xx_SYSTEM_INTERRUPT_CLEAR  0x015
#define VL6180xx_SYSTEM_FRESH_OUT_OF_RESET 0x016
#define VL6180xx_SYSTEM_GROUPED_PARAMETER_HOLD 0x017

#define VL6180xx_SYSRANGE_START          0x018
#define VL6180xx_SYSRANGE_THRESH_HIGH    0x019
#define VL6180xx_SYSRANGE_THRESH_LOW     0x01A
#define VL6180xx_SYSRANGE_INTERMEASUREMENT_PERIOD 0x01B
#define VL6180xx_SYSRANGE_MAX_CONVERGENCE_TIME 0x01C
#define VL6180xx_SYSRANGE_CROSSTALK_COMPENSATION_RATE 0x01E
#define VL6180xx_SYSRANGE_CROSSTALK_VALID_HEIGHT 0x021
#define VL6180xx_SYSRANGE_EARLY_CONVERGENCE_ESTIMATE 0x022
#define VL6180xx_SYSRANGE_PART_TO_PART_RANGE_OFFSET 0x024
#define VL6180xx_SYSRANGE_RANGE_IGNORE_VALID_HEIGHT 0x025
#define VL6180xx_SYSRANGE_RANGE_IGNORE_THRESHOLD 0x026
#define VL6180xx_SYSRANGE_MAX_AMBIENT_LEVEL_MULT 0x02C
#define VL6180xx_SYSRANGE_RANGE_CHECK_ENABLES 0x02D
#define VL6180xx_SYSRANGE_VHV_RECALIBRATE 0x02E
#define VL6180xx_SYSRANGE_VHV_REPEAT_RATE 0x031

#define VL6180xx_SYSALS_START             0x038
#define VL6180xx_SYSALS_THRESH_HIGH       0x03A
#define VL6180xx_SYSALS_THRESH_LOW        0x03C
#define VL6180xx_SYSALS_INTERMEASUREMENT_PERIOD 0x03E
#define VL6180xx_SYSALS_ANALOGUE_GAIN     0x03F
#define VL6180xx_SYSALS_INTEGRATION_PERIOD 0x040

#define VL6180xx_RESULT_RANGE_STATUS      0x04D
#define VL6180xx_RESULT_ALS_STATUS        0x04E
#define VL6180xx_RESULT_INTERRUPT_STATUS_GPIO 0x04F
```

```

#define VL6180xx_RESULT_ALS_VAL          0x050
#define VL6180xx_RESULT_HISTORY_BUFFER  0x052
#define VL6180xx_RESULT_RANGE_VAL       0x062
#define VL6180xx_RESULT_RANGE_RAW       0x064
#define VL6180xx_RESULT_RANGE_RETURN_RATE 0x066
#define VL6180xx_RESULT_RANGE_REFERENCE_RATE 0x068
#define VL6180xx_RESULT_RANGE_RETURN_SIGNAL_COUNT 0x06C
#define VL6180xx_RESULT_RANGE_REFERENCE_SIGNAL_COUNT 0x070
#define VL6180xx_RESULT_RANGE_RETURN_AMB_COUNT 0x074
#define VL6180xx_RESULT_RANGE_REFERENCE_AMB_COUNT 0x078
#define VL6180xx_RESULT_RANGE_RETURN_CONV_TIME 0x07C
#define VL6180xx_RESULT_RANGE_REFERENCE_CONV_TIME 0x080

#define VL6180xx_READOUT_AVERAGING_SAMPLE_PERIOD 0x10A
#define VL6180xx_FIRMWARE_BOOTUP 0x119
#define VL6180xx_FIRMWARE_RESULT_SCALER 0x120
#define VL6180xx_I2C_SLAVE_DEVICE_ADDRESS 0x212
#define VL6180xx_INTERLEAVED_MODE_ENABLE 0x2A3

#define VL6180xx_ADDRESS 0x29

const char *ssid = "HIT-FRITZBOX-7490";
const char *password = "63601430989011937932";

//const char *ssid = "DeepThought";
//const char *password = "1marvin@Beeblebrox1";

ESP8266WebServer server(80);

uint8_t distance; // value read from sensor
String webString=""; // String to display

void setup() {
  Serial.begin(115200); //Start Serial at 115200bps
  Wire.begin(); //Start I2C library
  delay(100); // delay .1s

  if(VL6180xxInit() != 0){
    Serial.println("FAILED TO INITIALIZE"); //Initialize device and check for errors
  };
  VL6180xxDefaultSettings(); //Load default settings to get started.
  delay(2000); // delay 1s

  // Wait for connection
  connectWiFi();

  server.begin();
  Serial.println("HTTP server started");

  server.on("/", handle_root);
  server.on("/distance", sendDistance);
}

void loop() {

```

```

//Get Distance and report in mm
Serial.print("Distance measured (mm) = ");
Serial.println(getDistance());
delay(1000);

server.handleClient();

Serial.println("");
Serial.println("*****");
Serial.println("Distance Reading Server");
Serial.print("Connected to ");
Serial.println(ssid);
Serial.print("IP address: ");
Serial.println(WiFi.localIP());
}

void connectWiFi() {
  // Connect to WiFi network
  WiFi.begin(ssid, password);
  Serial.print("\n\r\n\rWorking to connect");
  Serial.println("");
  Serial.println("Distance Reading Server");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  if (WiFi.status() == WL_CONNECTED) {
    return;
  }
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.println("no WiFi connection");
    delay(2000);
  }
}

void handle_root() {
  server.send(200, "text/plain", "Hello from the esp8266");
  delay(2000);
}

void sendDistance(){
  distance = getDistance(); // read sensor
  webString="Abstand: "+String(int(distance))+ " mm"; // Arduino has a hard time with float to string
  server.send(200, "text/plain", webString); // send to someones browser when asked
  delay(2000);
}

uint8_t VL6180xxInit(void){
  uint8_t data; //for temp data storage

  data = VL6180xx_getRegister(VL6180xx_SYSTEM_FRESH_OUT_OF_RESET);

```

```
if(data != 1) return VL6180xx_FAILURE_RESET;
```

```
VL6180xx_setRegister(0x0207, 0x01);
VL6180xx_setRegister(0x0208, 0x01);
VL6180xx_setRegister(0x0096, 0x00);
VL6180xx_setRegister(0x0097, 0xfd);
VL6180xx_setRegister(0x00e3, 0x00);
VL6180xx_setRegister(0x00e4, 0x04);
VL6180xx_setRegister(0x00e5, 0x02);
VL6180xx_setRegister(0x00e6, 0x01);
VL6180xx_setRegister(0x00e7, 0x03);
VL6180xx_setRegister(0x00f5, 0x02);
VL6180xx_setRegister(0x00d9, 0x05);
VL6180xx_setRegister(0x00db, 0xce);
VL6180xx_setRegister(0x00dc, 0x03);
VL6180xx_setRegister(0x00dd, 0xf8);
VL6180xx_setRegister(0x009f, 0x00);
VL6180xx_setRegister(0x00a3, 0x3c);
VL6180xx_setRegister(0x00b7, 0x00);
VL6180xx_setRegister(0x00bb, 0x3c);
VL6180xx_setRegister(0x00b2, 0x09);
VL6180xx_setRegister(0x00ca, 0x09);
VL6180xx_setRegister(0x0198, 0x01);
VL6180xx_setRegister(0x01b0, 0x17);
VL6180xx_setRegister(0x01ad, 0x00);
VL6180xx_setRegister(0x00ff, 0x05);
VL6180xx_setRegister(0x0100, 0x05);
VL6180xx_setRegister(0x0199, 0x05);
VL6180xx_setRegister(0x01a6, 0x1b);
VL6180xx_setRegister(0x01ac, 0x3e);
VL6180xx_setRegister(0x01a7, 0x1f);
VL6180xx_setRegister(0x0030, 0x00);
```

```
return 0;
```

```
}
```

```
void VL6180xxDefaultSettings(void){
```

```
//Recommended settings from datasheet
```

```
//Enable Interrupts on Conversion Complete (any source)
```

```
VL6180xx_setRegister(VL6180xx_SYSTEM_INTERRUPT_CONFIG_GPIO, (4 << 3)|(4) ); // Set GPIO1 high when
sample complete
```

```
VL6180xx_setRegister(VL6180xx_SYSTEM_MODE_GPIO1, 0x10); // Set GPIO1 high when sample complete
```

```
VL6180xx_setRegister(VL6180xx_READOUT_AVERAGING_SAMPLE_PERIOD, 0x30); //Set Avg sample period
```

```
VL6180xx_setRegister(VL6180xx_SYSALS_ANALOGUE_GAIN, 0x46); // Set the ALS gain
```

```
VL6180xx_setRegister(VL6180xx_SYSRANGE_VHV_REPEAT_RATE, 0xff); // Set auto calibration period (Max =
255)/(OFF = 0)
```

```
VL6180xx_setRegister(VL6180xx_SYSALS_INTEGRATION_PERIOD, 0x63); // Set ALS integration time to 100ms
```

```
VL6180xx_setRegister(VL6180xx_SYSRANGE_VHV_RECALIBRATE, 0x01); // perform a single temperature
calibration
```

```
//Optional settings from datasheet
```

```

VL6180xx_setRegister(VL6180xx_SYSRANGE_INTERMEASUREMENT_PERIOD, 0x09); // Set default ranging
inter-measurement period to 100ms
VL6180xx_setRegister(VL6180xx_SYSALS_INTERMEASUREMENT_PERIOD, 0x0A); // Set default ALS inter-
measurement period to 100ms
VL6180xx_setRegister(VL6180xx_SYSTEM_INTERRUPT_CONFIG_GPIO, 0x24); // Configures interrupt on 'New
Sample Ready threshold event'
//Additional settings defaults from community
VL6180xx_setRegister(VL6180xx_SYSRANGE_MAX_CONVERGENCE_TIME, 0x32);
VL6180xx_setRegister(VL6180xx_SYSRANGE_RANGE_CHECK_ENABLES, 0x10 | 0x01);
VL6180xx_setRegister(VL6180xx_SYSRANGE_EARLY_CONVERGENCE_ESTIMATE, 0x7B );
VL6180xx_setRegister(VL6180xx_SYSALS_INTEGRATION_PERIOD, 0x64);

VL6180xx_setRegister(VL6180xx_READOUT_AVERAGING_SAMPLE_PERIOD,0x30);
VL6180xx_setRegister(VL6180xx_SYSALS_ANALOGUE_GAIN,0x40);
VL6180xx_setRegister(VL6180xx_FIRMWARE_RESULT_SCALER,0x01);
}

```

```

uint8_t getDistance()
{
    VL6180xx_setRegister(VL6180xx_SYSRANGE_START, 0x01); //Start Single shot mode
    delay(10);
    return VL6180xx_getRegister(VL6180xx_RESULT_RANGE_VAL);

    // distance = VL6180xx_getRegister(VL6180xx_RESULT_RANGE_VAL);
    // return distance;

    VL6180xx_setRegister(VL6180xx_SYSTEM_INTERRUPT_CLEAR, 0x07);
    //return distance;
}

```

```

uint8_t VL6180xx_getRegister(uint16_t registerAddr)
{
    uint8_t data;

    Wire.beginTransmission( VL6180xx_ADDRESS ); // Address set on class instantiation
    Wire.write((registerAddr >> 8) & 0xFF); //MSB of register address
    Wire.write(registerAddr & 0xFF); //LSB of register address
    Wire.endTransmission(false); //Send address and register address bytes
    Wire.requestFrom( VL6180xx_ADDRESS , 1);
    data = Wire.read(); //Read Data from selected register

    return data;
}

```

```

void VL6180xx_setRegister(uint16_t registerAddr, uint8_t data)
{
    Wire.beginTransmission( VL6180xx_ADDRESS ); // Address set on class instantiation
    Wire.write((registerAddr >> 8) & 0xFF); //MSB of register address
    Wire.write(registerAddr & 0xFF); //LSB of register address
    Wire.write(data); // Data/setting to be sent to device.
    Wire.endTransmission(); //Send address and register address bytes
}

```


II. Code in Intel XDK

```

var glo=0;
var formIP;
/*jshint browser:true */
/*global $ */(function()
{
  "use strict";
  /*
   hook up event handlers
  */
  function sc()
  {
    window.open(formIP, "_system");
    document.addEventListener("intel.xdk.device.remote.close",function() {glo=0;}, false);

    if(glo==1)
    {
      window.setTimeout(sc,3000);
    }
    else{}
  }

  function register_event_handlers()
  {

    /* button Start */
    $(document).on("click", ".uib_w_1", function(evt)
    {
      /* your code goes here */
      glo=1;
      formIP= document.getElementById("input").value;
      window.setTimeout(sc,1000);

    });

    /* button Stop */
    $(document).on("click", ".uib_w_2", function(evt)
    {
    });

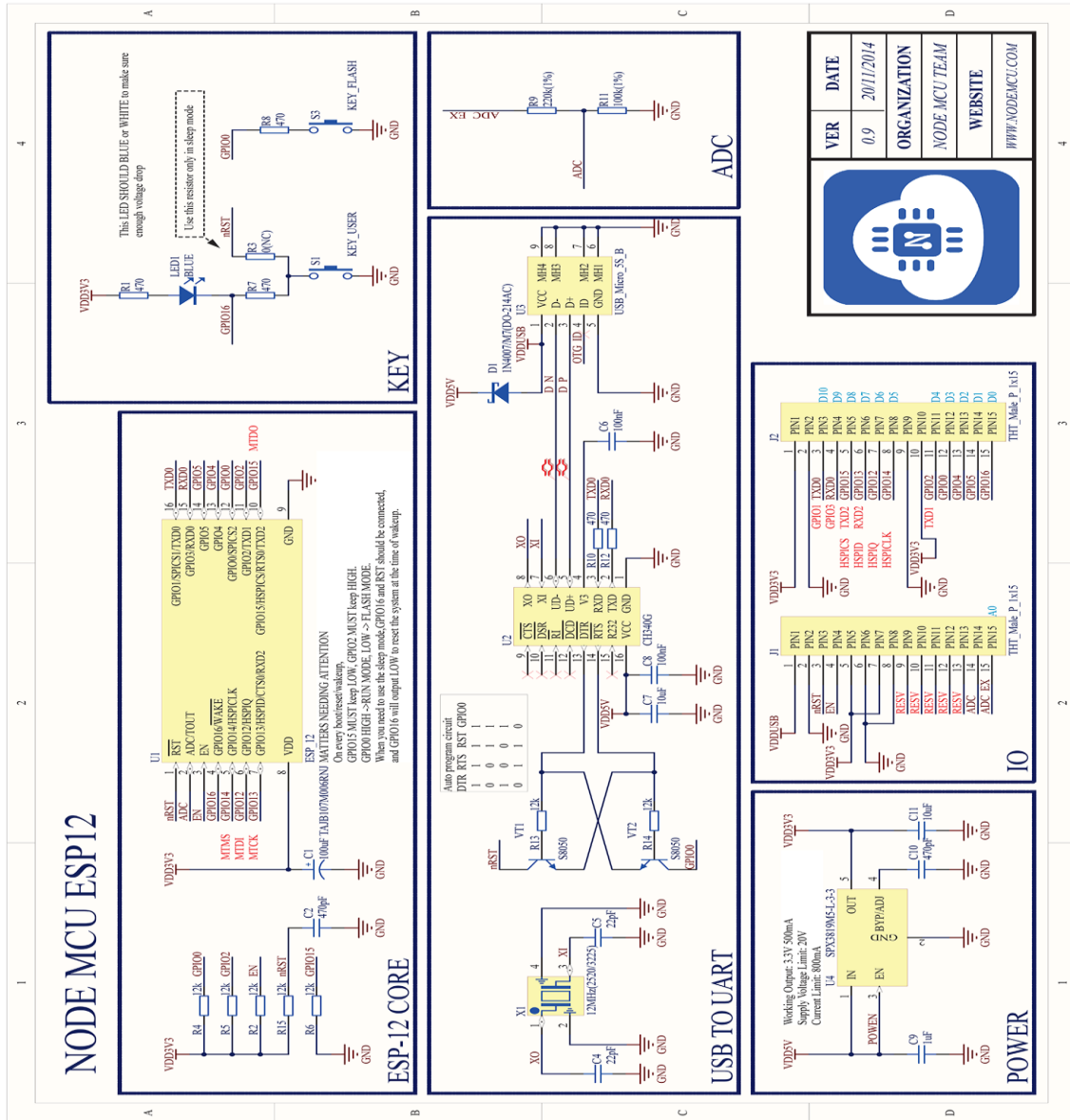
    /* button Button */
    $(document).on("click", ".uib_w_4", function(evt)
    {
      /* your code goes here */
      navigator.app.exitApp();
    });

  }

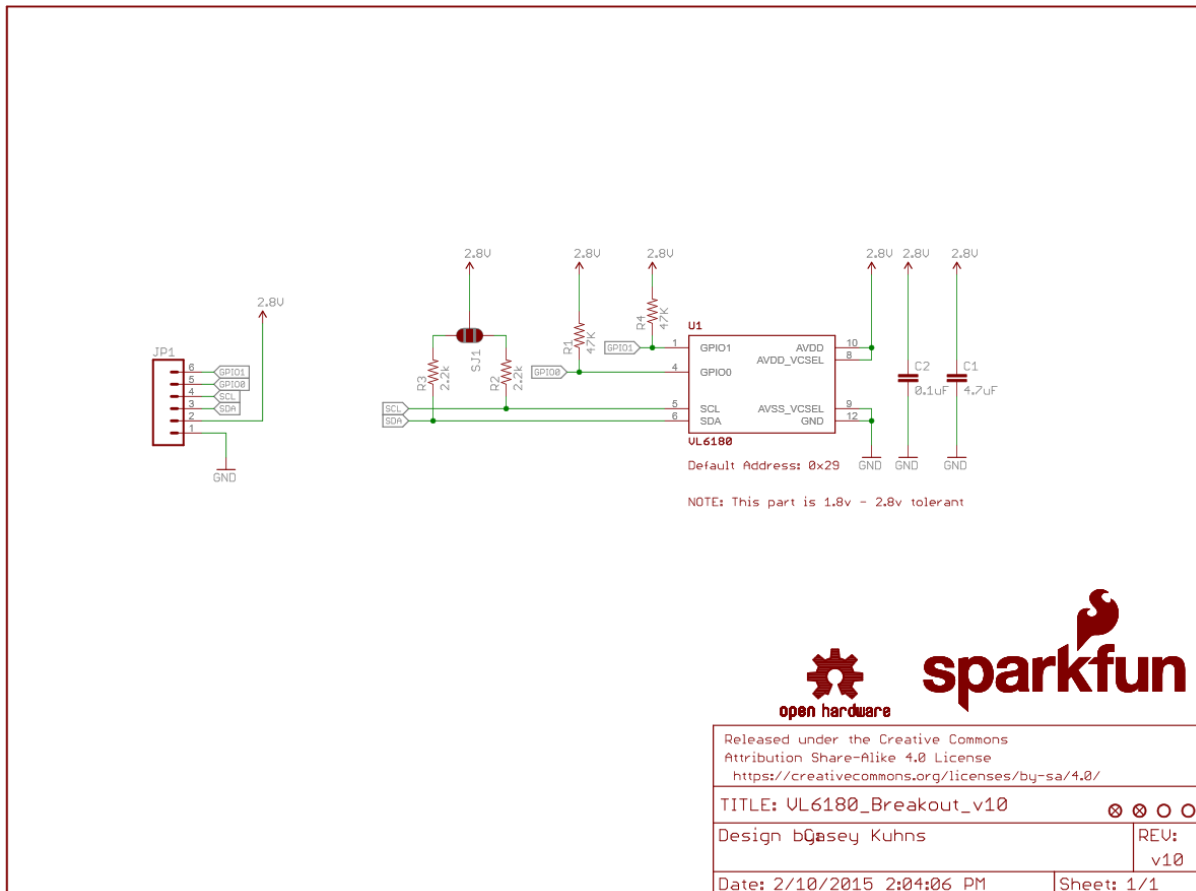
  document.addEventListener("app.Ready", register_event_handlers, false);
})();

```

III. Schaltplan: NodeMCU 1.0(ESP-12E Module)



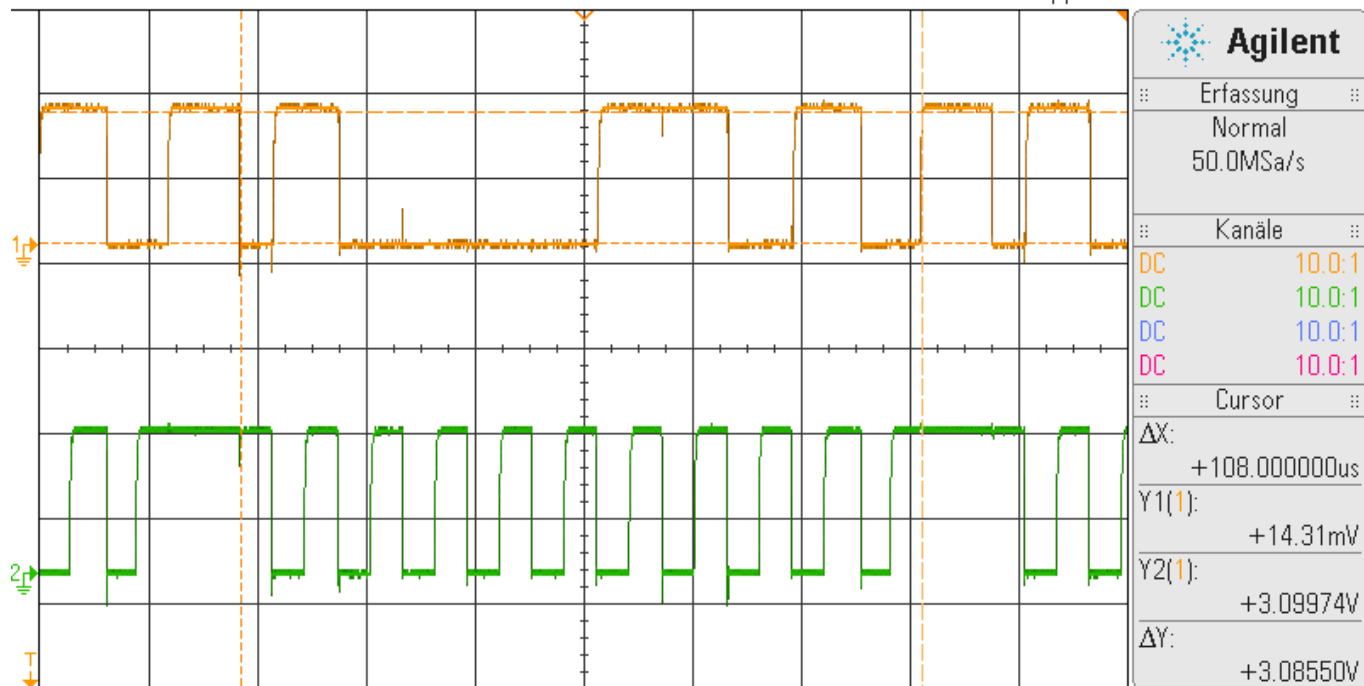
Iv. Schaltplan: VL 6180 Sensor



V. I2C

DSO-X 2014A, MY54101096: Tue Apr 19 11:58:47 2016

1 2.00V/ 2 2.00V/ 3 4 -271.6% 17.20%/ Stopp f 1 -13.6V



Menü "Cursor"

Modus Signalverfolgung	X1-Kanal 1	X2-Kanal 1	Cursor X2	Einh. ↓	X1: -326.000000us X2: -218.000000us
---------------------------	---------------	---------------	--------------	------------	--