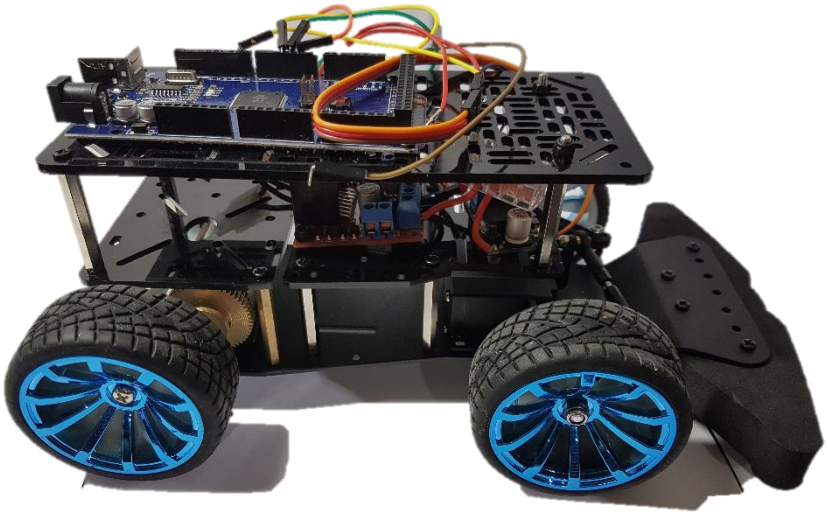
MINT-Camp
Autonomes Fahren

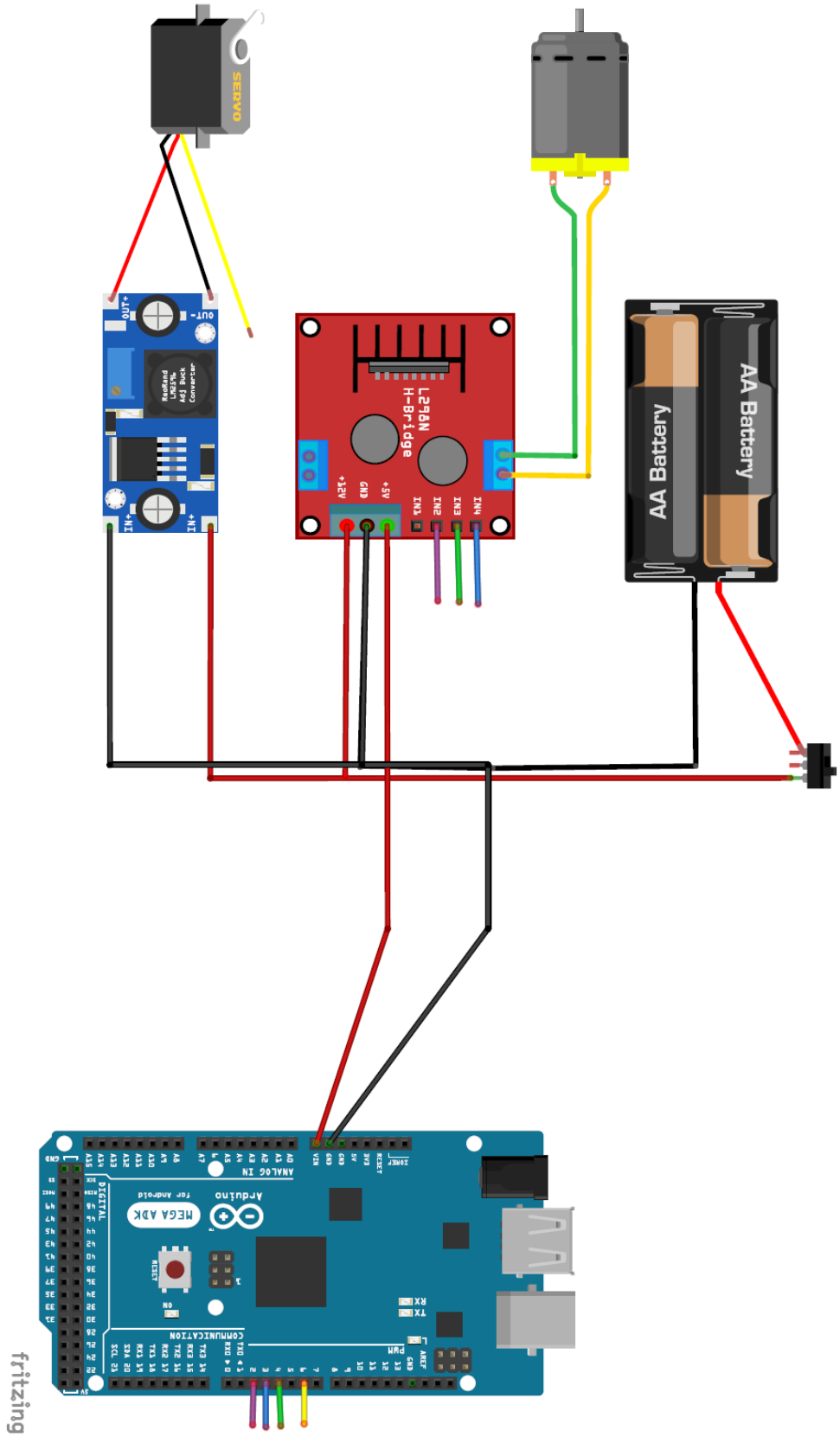


Inhaltsverzeichnis

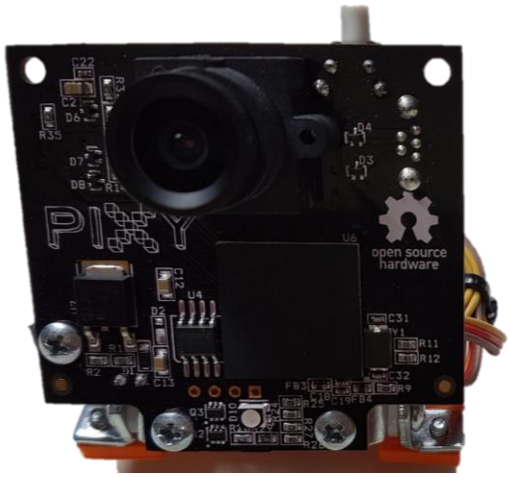
Fachliches Konzept MINT Camp

Tag	Zeit	Inhalte	
Dienstag	14.00-14.30	Ankunft	
	14.30-15.00	Begrüßung und Impulsvortrag	
	15.00-16.30	Einführungsworkshop Arduino (Mikrocontroller) und Robotik Impulsvortrag (kurz)	Einführungsworkshop SolidEdge, Fräse und 3D-Drucker
	16.30-18.00	Einführungsworkshop SolidEdge, Fräse und 3D-Drucker	Einführungsworkshop Arduino (Mikrocontroller) und Robotik Impulsvortrag (kurz)
Mittwoch	09.00-09.15	Einteilung der vier Gruppen (Steuerung, Objekterkennung über Kamera, Lidar Laser, Ultraschall und Infrarot)	
	09.15-12.00	Einarbeitung in Themen und erste Implementierung	
	12.00-13.00	Mittagessen	
	13.00-14.00	Fachvortrag Hochschule Hof - Autonomes Fahren/Künstliche Intelligenz	
	14.00-18.00	Weiteres Arbeiten an und mit Sensoren Modellieren und Drucken/Fräsen von Halterungen für Sensoren Implementierung	
Donnerstag	09.00-13.00	Ausflug zu regionalem Unternehmen aus dem Bereich Autonomes Fahren	
	14.00-18.00	Zusammenführung der Gruppenergebnisse an einem Modell Testen und Entwicklung von Lösungen	
Freitag	09.00-14.00	Weiteres Testen des Modells Ggf. Einbau eines GPS-Moduls und Lösen der Abschlussaufgabe von Punkt A zu Punkt B inkl. Ausweichen bei statischen und dynamischen Hindernissen	

Schaltplan


SERVO
OUT+
OUT-
IN+
IN-
L298N
H-Bridge
+12V
GND
+5V
IN1
IN2
IN3
IN4
AA Battery
AA Battery
ANALOG IN
DIGITAL
COMMUNICATION
MEGA ADK
Arduino
for Android
fritzing

Pixy Cam



Beschreibung:

Die Pixy Cam ist eine Kamera mit Mikrocontroller zum Verarbeiten der Bilddaten. Sie kann mit zwei Servo Motoren verbunden werden um ein Objekt zu verfolgen. Über das Programm PixyMon können Einstellungen vorgenommen und die Kamera getestet werden.

Was kann der Sensor:

Die Pixy Cam kann Objekte die sich in ihrem Bildbereich befinden verfolgen und deren Position an den Arduino liefern. Dabei verwendet die Pixy Cam einen schnellen, robusten, farbbasierten Filteralgorithmus (Color Code) zur Erkennung von Objekten. Pixy berechnet die Farbe (Farbton) und Sättigung jedes RGB-Pixels aus dem Bildsensor und verwendet diese als primären Filterparameter. Das Erkennungssystem verwendet einen internen Komponenten-Algorithmus, um zu bestimmen, wo ein Objekt beginnt und das andere aufhört. Pixy kompiliert dann die Größen und Positionen der einzelnen Objekte. Die Informationen werden über UART, SPI oder I2C an den Arduino geschickt. Dieser kann aus den Positionsinformationen die Fahrmanöver für den Roboter planen und ausführen.

Dokumentation:

siehe:

- https://docs.pixycam.com/wiki/doku.php?id=wiki:v1:start
- https://5volt-junkie.net/pixy-arduino/

Beispielprogramm Arduino:

```
#include <SPI.h>
#include <Pixy.h>

// This is the main Pixy object
Pixy pixy;

void setup()
{
 Serial.begin(9600);
 Serial.print("Starting...\n");
 pixy.init();
}

void loop()
{
 static int i = 0;
 int j;
 uint16_t blocks;
 char buf[32];

 // grab blocks!
 blocks = pixy.getBlocks();

 // If there are detect blocks, print them!

 if (blocks)
 { i++;
  // do this (print) every 50 frames because printing every frame would bog down the Arduino
  if (i%50==0)
  {
   sprintf(buf, "Detected %d:\n", blocks);
   Serial.print(buf);
   for (j=0; j<blocks; j++)
   {
    sprintf(buf, "  block %d: ", j);
    Serial.print(buf);
    pixy.blocks[j].print();
   }
  }
 }
}
```

Pixy Cam V2

Beschreibung:

Die Pixy Cam V2 ist eine verbesserte Version der ersten Pixy Cam und verfügt über alle Features der ersten Version und wir um zusätzliche Algorithmen erweitert. Die Pixy Cam V2 wird mit dem Programm PixyMon2 angesteuert und kann nicht mit PixyMon1 zusammenarbeiten. Zusätzlich kann die Kamera aber auch Linien erkennen, Kreuzungen von Linien erfassen und Barcodes auslesen.

Was kann der Sensor:

Die Pixy Cam kann wie die erste Version Objekte die sich in ihrem Bildbereich befinden verfolgen und deren Position an den Arduino liefern. Zusätzlich kann die Kamera aber auch Linien erkennen, Kreuzungen von Linien erfassen und Barcodes auslesen. Die Verarbeitungsrate hat sich auf 60 fps verbessert und liefert so noch zuverlässigere Werte an den Arduino. Die Informationen werden über UART, SPI oder I2C an den Arduino geschickt, die bereitgestellte Bibliothek wurde vereinfacht und kann ausschließlich für die Pixy Cam V2 genutzt werden. Es wurde zusätzlich zwei weiße LEDs angebracht, die zu Beleuchtung von Objekten genutzt werden können. Wird die Kamera zur Linienverfolgung eingesetzt, liefert der Sensor Informationen über die Richtung der zu folgenden Linie, andere Linien, und Kreuzungspunkte von Linien. Dabei sollte beachtet werden, dass in der Software voreingestellt werden muss, ob der Roboter helle Linien auf einem dunklen Untergrund folgt oder dunkle Linien auf einem hellen Untergrund.

Dokumentation:

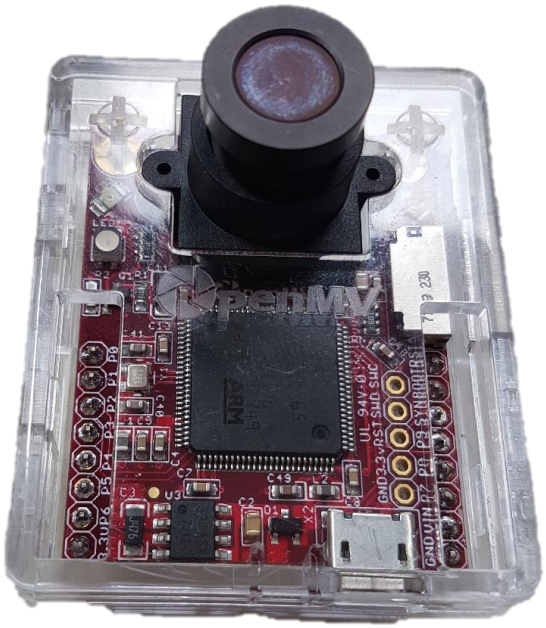
siehe:

- https://docs.pixycam.com/wiki/doku.php?id=wiki:v2:overview
- https://docs.pixycam.com/wiki/doku.php?id=wiki:v2:start

Beispielprogramm Arduino:

```
#include <Pixy2.h>

Pixy2 pixy;

void setup()
{
  Serial.begin(115200);
  Serial.print("Starting...\n");

  pixy.init();
  // change to the line_tracking program.  Note, changeProg can use partial strings, so for example,
  // you can change to the line_tracking program by calling changeProg("line") instead of the whole
  // string changeProg("line_tracking")
  Serial.println(pixy.changeProg("line"));
}

void loop()
{
  int8_t i;
  char buf[128];

  pixy.line.getMainFeatures();

  if (pixy.line.numVectors)
    pixy.line.vectors->print();

  if (pixy.line.numIntersections)
    pixy.line.intersections->print();

  if (pixy.line.barcodes)
    pixy.line.barcodes->print();

}
```

OpenMV Cam M7



Beschreibung:

Die OpenMV Cam ist eine Kamera, die mit einem Mikrocontroller verbunden ist, der die Kamerabilder effizient verarbeitet. Das System wird mit der Programmiersprache Python programmiert. Dabei reichen jedoch einige wenige Programmierbefehle um beeindruckende Resultate zu erzielen. Zur Programmierung wird die Entwicklungsumgebung openMV IDE verwendet, die bereits viele Beispiele zur Programmierung der Cam mitliefert.

Was kann der Sensor:

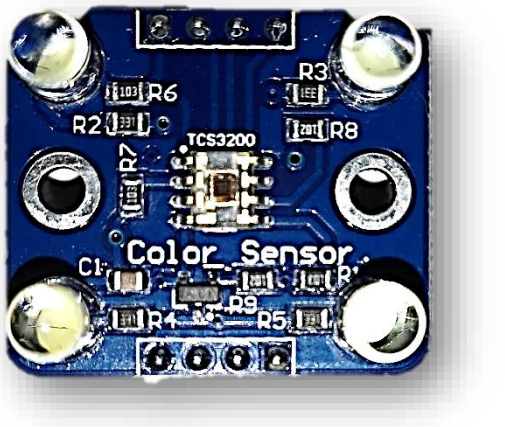
Die OpenMV Cam ist vielseitig einsetzbar. Sie verarbeitet Kamerabilder sehr schnell und kann zur Linienerkennung eingesetzt werden (siehe Beispiele 09-Feature-Detection -> find_lines oder -> linear_regression_robust). Die Kamera kann aber auch QR-Codes, Tags und Barcodes lesen. So können zum Beispiel Verkehrszeichen gelesen werden (siehe Beispiele 16-Codes). Auch Farbcodes oder grundsätzlich Farben können erkannt und verfolgt werden(siehe Beispiele 10-Color-Tracking).Zusätzlich ist es möglich Bilder von der Fahrt aufzunehmen und auf einer Mini-SD-Karte zu speichern (siehe Beispiele 05-Snapshot und 06-Video-Recording).
Die Ausgewerteten Informationen können von der Kamera per I2C oder UART an den Arduino geschickt werden, der dann die Steuerung der Motoren übernimmt.

Dokumentation:

siehe:

- http://docs.openmv.io/openmvcam/quickref.html
- http://docs.openmv.io/openmvcam/tutorial/index.html

TCS34725/ TCS3200 Farbsensor



Beschreibung:

Der TCS34725/ TCS3200 Farbsensor misst Farbwerte und die Farbtemperatur des Umgebungslichts. Der Sensor TCS34725 wird per I2C an den Arduino angeschlossen und liefert mit der entsprechenden Bibliothek die Farbwerte für rotes, blaues und grünes Licht, sowie die Helligkeit und die Farbtemperatur. Auf dem Sensormodul ist eine weiße LED eingebaut, die über den LED-Pin auch ausgeschaltet werden kann. Der TCS3200 liefert über die vier Datenpins die Information über den Farbwert.

Was kann der Sensor:

Der Sensor kann verwendet werden um die Farbe von Objekten zu erkennen. So kann er zum Beispiel farbige Linien auf der Fahrbahn erkennen. Auch zur Erfassung der Umgebungsbedingungen (Helligkeit der Umgebung) kann der Sensor eingesetzt werden.

Dokumentation:

siehe:

- https://learn.adafruit.com/adafruit-color-sensors/arduino-code (TCS34725)
- https://www.arduino.cc/en/Reference/Wire
- https://funduino.de/nr-07-farbsensor-am-arduino (TCS3200)
- https://howtomechatronics.com/tutorials/arduino/arduino-color-sensing-tutorial-tcs230-tcs3200-color-sensor/ (TCS32000)

Beispielprogramm Arduino:

```
#include <Wire.h>
#include "Adafruit_TCS34725.h"

/* Initialise with default values (int time = 2.4ms, gain = 1x) */
// Adafruit_TCS34725 tcs = Adafruit_TCS34725();

/* Initialise with specific int time and gain values */
Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_700MS,
TCS34725_GAIN_1X);

void setup(void) {
  Serial.begin(9600);

  if (tcs.begin()) {
    Serial.println("Found sensor");
  } else {
    Serial.println("No TCS34725 found ... check your connections");
    while (1);
  }

  // Now we're ready to get readings!
}

void loop(void) {
  uint16_t r, g, b, c, colorTemp, lux;

  tcs.getRawData(&r, &g, &b, &c);
  colorTemp = tcs.calculateColorTemperature(r, g, b);
  lux = tcs.calculateLux(r, g, b);

  Serial.print("Color Temp: "); Serial.print(colorTemp, DEC); Serial.print(" K - ");
  Serial.print("Lux: "); Serial.print(lux, DEC); Serial.print(" - ");
  Serial.print("R: "); Serial.print(r, DEC); Serial.print(" ");
  Serial.print("G: "); Serial.print(g, DEC); Serial.print(" ");
  Serial.print("B: "); Serial.print(b, DEC); Serial.print(" ");
  Serial.print("C: "); Serial.print(c, DEC); Serial.print(" ");
  Serial.println(" ");
}
```

RPLIDAR A2M8-R4

Beschreibung:

Das RPLIDAR ist ein Sensor, der optisch den Abstand zu Objekten über einen Laser misst. Dabei werden Laserimpulse ausgesendet und das reflektierte Licht gemessen. Aus der Lichtlaufzeit wird die Entfernung zu einem Objekt berechnet. Der Sensor dreht sich während der Objekterfassung laufend um 360° und tastet somit die gesamte Umgebung des Roboters ab. Die Auflösung der Umgebung hängt dabei von der Rotationsgeschwindigkeit ab.

Was kann der Sensor:

Der Sensor misst fortlaufend die Distanz zu Objekten der Umgebung. Der Sensor liefert neben der Entfernung des Objekts auch den Winkel des ausgesendeten Laserstrahls. So kann ein Abbild der Umgebung gemacht werden um Rückschlüsse zu ziehen, ob der Roboter vor einem Hindernis steht oder nicht und welche Wege zum Umfahren des Hindernisses zur Verfügung stehen. Der Sensor kann über das mitgelieferte SDK getestet werden. Dabei kann man sehr gut die gelieferten Daten interpretieren. Zur Steuerung des Roboters muss der Sensor über UART mit dem Arduino verbunden werden. Zum Auslesen der Sensordaten gibt es die Arduino-Bibliothek RPLidar.

Dokumentation:

siehe:

- https://www.robotshop.com/media/files/pdf2/ld208_slamtec_rplidar_datasheet_a2m8_v1.1_en_2_.pdf
- https://www.robotshop.com/media/files/pdf2/rpk-02-user-manual.pdf
- https://www.robotshop.com/media/files/pdf2/rpk-02-communication-protocol.pdf
- https://www.robotshop.com/media/files/pdf2/rpk-02-sdk-manual.pdf
- https://www.robotshop.com/en/rplidar-a2m8-360-laser-scanner.html

Beispielprogramm Arduino:

```
// This sketch code is based on the RPLIDAR driver library provided by RoboPeak
#include <RPLidar.h>

// You need to create an driver instance
RPLidar lidar;

#define RPLIDAR_MOTOR 3 // The PWM pin for control the speed of RPLIDAR's motor.
                        // This pin should connected with the RPLIDAR's MOTOCTRL signal

void setup() {
  // bind the RPLIDAR driver to the arduino hardware serial
  lidar.begin(Serial);

  // set pin modes
  pinMode(RPLIDAR_MOTOR, OUTPUT);
}

void loop() {
  if (IS_OK(lidar.waitPoint())) {
    float distance = lidar.getCurrentPoint().distance; //distance value in mm unit
    float angle    = lidar.getCurrentPoint().angle; //anglue value in degree
    bool  startBit = lidar.getCurrentPoint().startBit; //whether this point is belong to a new scan
    byte  quality  = lidar.getCurrentPoint().quality; //quality of the current measurement

    //perform data processing here...


  } else {
    analogWrite(RPLIDAR_MOTOR, 0); //stop the rplidar motor

    // try to detect RPLIDAR...
    rplidar_response_device_info_t info;
    if (IS_OK(lidar.getDeviceInfo(info, 100))) {
      // detected...
      lidar.startScan();

      // start motor rotating at max allowed speed
      analogWrite(RPLIDAR_MOTOR, 255);
      delay(1000);
    } }}
```

Leddarone Sensor Module UART

Beschreibung:

Das Leddarone Sensor Module von Laddar Tech ist ein Time-of-Flight Sensor, der die Entfernung von Objekten über die Dauer die ein Lichtimpuls von der Quelle über die Reflektion am Objekt bis zum Empfänger benötigt. Das Modul kann mit großen Entfernungen arbeiten und arbeitet auf 5cm genau. Es liefert die Entfernung von Objekten über die Serielle Schnittstelle.

Was kann der Sensor:

Der Sensor misst fortlaufend die Distanz zu Objekten der Umgebung. Der Sensor liefert neben der Entfernung des Objekts einen Wert, der darüber Aufschluss gibt, wie zuverlässig die Information ist. Zur Steuerung des Roboters muss der Sensor über UART mit dem Arduino verbunden werden. Zum Auslesen der Sensordaten gibt es die Arduino-Bibliothek <Leddar.h>.

Dokumentation:

siehe:

- https://leddartech.com/app/uploads/dlm_uploads/2018/04/Spec-Sheets-LeddarOne-ENG-12avril2018-web.pdf
- https://www.robotshop.com/de/de/leddartech-leddarone-optischer-entfernungsmesser-33v-uart.html
- https://www.robotshop.com/media/files/pdf2/rb-led-02-04_-_54a0025-2_leddar_one_user_guide_-_2016-09-01.pdf
- https://www.robotshop.com/media/files/pdf2/application_note_-_leddar_proximity_detection_and_distance_measurement_for_manufacturing_conveyor_system.pdf
- https://www.robotshop.com/media/files/images2/overview-of-a-novel-led-based-detection-and-ranging-technology.pdf

Beispielprogramm Arduino:

```
#include <SoftwareSerial.h>
#include <Leddar.h>

LeddarOne Leddar1(115200,1);
//Baudrate = 115200
//Modbus slave ID = 01
// NOTE: If your RS-485 shield has a Tx Enable (or DE) pin,
// use: Leddar Leddar1(115200,1, TxEnablePinNumber, 1);

void setup()
{
	Serial.begin(115200); //Opens serial connection at 115200bps.


	//Initialize Leddar
	Leddar1.init();
}

void loop()
{
	unsigned int Distance = 0;
	unsigned int Amplitude = 0;

	char result = Leddar1.getDetections();
	if (result >= 0)
	{
		// Show the first detection only
		Distance = Leddar1.Detections[0].Distance;
		Amplitude = Leddar1.Detections[0].Amplitude;

		Serial.print("Distance: "); Serial.println(Distance);
		Serial.print("Amplitude: "); Serial.println(Amplitude);
	}
	else
	{
		Serial.print("Error: "); Serial.println((int)result);
	}
	delay(50);}
```

Ultraschall Sensoren

Beschreibung:

Mit den Ultraschall Sensor Modulen HC-SR04 und URM V4.0 kann die Distanz zu Objekten in der Umgebung gemessen werden. Dabei kann der URM V4.0 zusätzlich zur Distanz, die auf drei verschiedene Arten ermittelt werden kann auch die Temperatur messen. Beide Sensoren ermitteln die Distanz zu Objekten über einen Schallimpuls, der von Objekten zurückgeworfen wird. Aus der Zeit zwischen dem Entsenden des Signals und dem Empfangen wird die Entfernung des Objekts ermittelt.

Was kann der Sensor:

Die Sensoren können Entfernungen zu anderen Objekten im Schallbereich in Entfernungen zwischen 5cm und 500cm abschätzen. Dabei verbreiten sich die Schallwellen trichterförmig vom Emitter aus. Mögliche Einflussgrößen auf die Messung nehmen dabei Absorption, Reflexion, Brechung, Streuung und Beugung der Schallwellen. Beide Sensoren verwenden zur Distanzmessung die Arduinoeigene Funktion pulseIn(pin, value). Die Sensoren können in vielen Szenarien verwendet werden: beim autonomen Einparken, zum Folgen von Fahrzeugen, zum Ausweichen von Hindernissen oder als Radar, wenn der Sensor auf einem rotierenden Servomotor angebracht wurde.

Dokumentation:

siehe:

- https://www.arduino.cc/reference/en/language/functions/advanced-io/pulsein/
- https://www.dfrobot.com/wiki/index.php/URM37_V4.0_Ultrasonic_Sensor_(SKU:SEN0001)
- https://www.roboter-bausatz.de/media/pdf/33/49/91/HC-SR04-Datasheet.pdf
- https://funduino.de/nr-10-entfernung-messen

Beispielprogramm Arduino:

```
int trigger=7;
int echo=6;
long dauer=0;
long entfernung=0;
void setup()
{
Serial.begin (9600);
pinMode(trigger, OUTPUT);
pinMode(echo, INPUT);
}
void loop()
{
digitalWrite(trigger, LOW);
delay(5);
digitalWrite(trigger, HIGH);
delay(10);
digitalWrite(trigger, LOW);
dauer = pulseIn(echo, HIGH);
entfernung = (dauer/2) * 0.03432;
if (entfernung >= 500 || entfernung <= 0)
{
Serial.println("Kein Messwert");
}
else
{
Serial.print(entfernung);
Serial.println(" cm");
}
delay(1000);
}
```

Indoor GPS

4 stationäre Beacons und
1 mobiler Beacon

Modem

Beschreibung:

Das Marvelmind Indoor Navigation System ist ein serienmäßiges Indoor-Navigationssystem, das zum Beispiel autonomen Robotern präzise Positionsdaten (±2cm) zur Verfügung stellt. Es wird auch verwendet, um Objekte zu verfolgen, die das Mobilteil (mobile beacon) installiert haben.

Was kann der Sensor:

Das Indoor GPS liefert Echtzeitdaten von den eingesetzten Beacons (stationär und mobil) entweder über das mitgelieferte Modem (angeschlossen an den Computer) oder direkt an den Arduino (angeschlossen an den mobilen Beacon).

Die Reichweite zwischen den stationären Beacons beträgt maximal 50 Meter, so dass mit dem Starterset eine Fläche von bis zu 1000m^2 abgedeckt werden kann.

Alle Beacons haben einen eingebauten Lithium-Ionen-Akku, so dass sie ohne externe Stromquelle eingesetzt werden können. Die Laufzeit hängt von der Nutzung ab und beträgt mindestens 72h (stationär) und 12h (mobil).

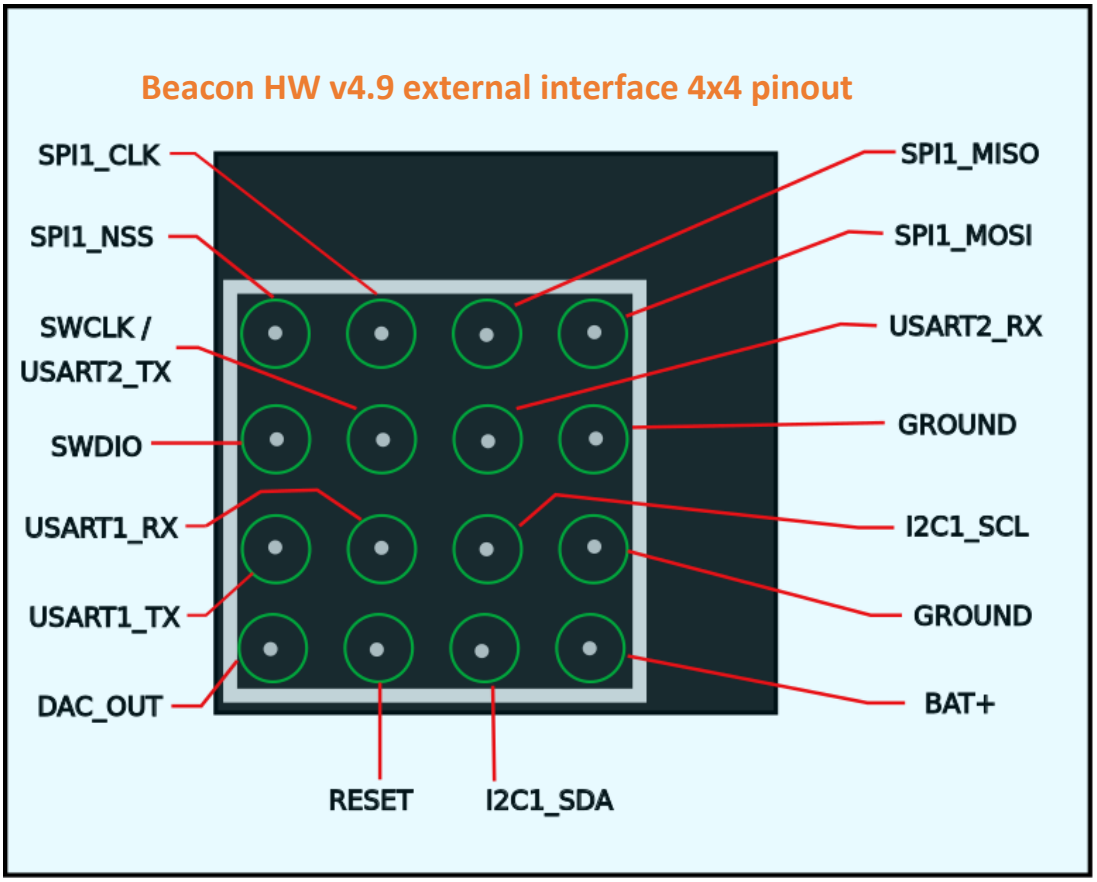
Der mobile Beacon kann die Daten über UART an den Arduino liefern.

Dokumentation:

siehe:

- https://marvelmind.com/wp-content/uploads/2017/08/marvelmind_hedgehog2arduino_interface_v2016_03_07.pdf
- https://marvelmind.com/download/
- https://www.marvelmind.com/pics/marvelmind_beacon_interfaces_v2017_09_13.pdf

Pinout Schema „Mobile Beacon“ an Arduino:

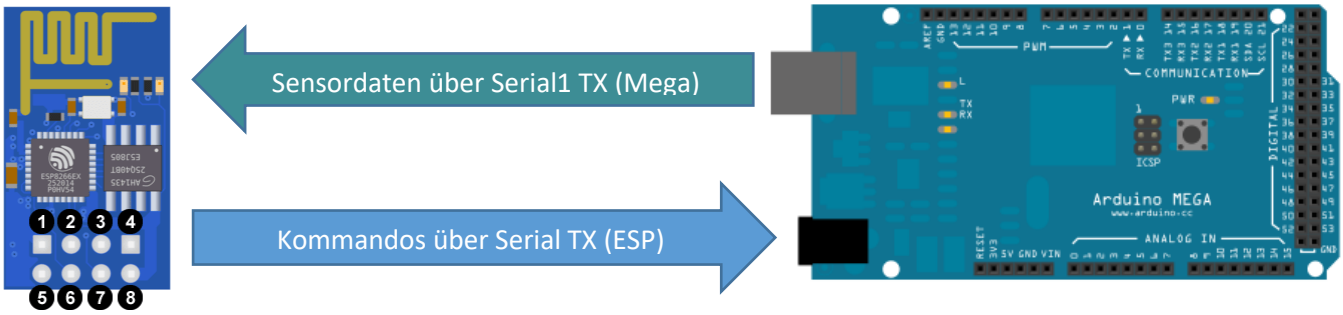


Beispielprogramm Arduino:

```
void setup()
{
 setup_hedgehog();//    Marvelmind hedgehog support initialize
}

void loop()
{
  bool showPath;
  delayMicroseconds(500);
  loop_hedgehog();// Marvelmind hedgehog service loop
  showPath= ((moveItemsNum != 0) && (digitalRead(READY_RECEIVE_PATH_PIN)== LOW));
  if (hedgehog_pos_updated)
   {// new data from hedgehog available
     hedgehog_pos_updated= 0;// clear new data flag
     if (!showPath)
      {
        Serial.println("X=");
        Serial.print(hedgehog_x);
        Serial.print("   ");
      }
     Serial.println("Y=");
```

```
    Serial.print(hedgehog_y);
    Serial.print("   ");
    Serial.println("Z=");
    Serial.print(hedgehog_z);
    Serial.print("  ");
   }

 if (showPath)
   {
     unsigned long newMillis= millis();
     if (newMillis>prevMoveItemShowTime+1000)
      {
        serial_show_move_item();
        moveItemShowIndex= (moveItemShowIndex + 1)%moveItemsNum;
        prevMoveItemShowTime= newMillis;
      }
   }
}
```

Arduino – Verbindung mit dem ESP8266



ESP8266

Arduino Mega

Beschreibung der Komponenten:

- ESP8266:
  - Fungiert als Webserver und liefert Webseiten an Clients. Die IP-Adresse des Servers ist standardmäßig 192.168.4.1. Wird diese Adresse im Browser aufgerufen Liefert der Webserver auf dem ESP eine HTML-Webseite aus.
  - Fungiert zusätzlich als Accesspoint und stellt ein WLAN zur Verfügung (MINTCAMP…). Bevor man mit dem Webserver Kontakt aufnehmen kann muss man sich im WLAN des Accesspoint anmelden.
  - Empfängt Werte über den Pin RX über Serial.read() vom Arduino Mega und baut diese in die Website ein
  - Sendet Benutzereingaben an den Arduino über Serial.write(). Diese Eingaben bekommt der ESP über die GET-Parameter der URL mit der der Client die Website aufruft mitgeteilt. (Beispiel: 192.168.4.1?power="off" -> Parameter power hat den Wert off)

- Arduino Mega:
  - Sammelt Sensordaten über seine digitalen und Analogen Pins und schickt diese über den Pin 18 (TX1) per Serial1.write() an den ESP.
  - Bekommt die Informationen des ESP über den Pin 19 (RX1) per Serial1.readBytes(). Diese Informationen müssen ausgewertet und für die Steuermanöver weiterverarbeitet werden.

Beschreibung des Quelltextes:

HTML-Dokument das vom ESP-Webserver angezeigt wird:

```
<head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1">
```

Hier wird HTML5 verwendet um die Seite auch auf dem Handy korrekt anzuzeigen, es sollten keine Anpassungen nötig sein.

```
<style>
      body {
          font-family:Palatino Linotype;
      }

      a:link, a:visited {
      color: white;
      padding: 10px 25px;
      text-align: center;
      text-decoration: none;
      display: inline-block;
      border-radius: 25%;
      }
  </style>
```

Hier werden per CSS Formatierungen festgelegt. Unter a:link, a:visited werden Links so verändert, dass sie wie Button aussehen.

```
<div align="center" >
      <h3>Power on and off</h3>
      <a style="background-color:#31B404" href="/?power=on">on</a>
      <a style="background-color:#DF0101" href="/?power=off">off</a>
      <br>
 </div>
```

Hier werden Links auf die Website eingefügt. Die Links verweisen auf sich selbst (href="/…"), jedoch wird der GET-Parameter power mit einem Wert belegt (?power=<*Wert*>). Dieser Parameter kann vom Webserver ausgewertet werden um entsprechende Kommandos an den Arduino zu schicken. Hier können bei Bedarf auch weitere Links mit anderen Parametern eingefügt werden, oder weitere Werte für den Parameter power. Die Werte on bzw. off sind willkürlich gewählte Texte.

```
<div align="center">
      <h3>Arduino-Wert</h3>
      Wert = <span id="wert"></span>
      <br>
 </div>
```

Hier wird der Bereich zum Anzeigen von Werten des Arduinos erstellt. `<span id="wert"></span>` versieht das Tag span mit einer ID auf die JavaScript zugreifen kann um zwischen dem Start- und dem Endtag einen Wert zu schreiben.

```
<script>
      setInterval(function(){getData();}, 500);
```

Mit dem `<script>` Tag beginnt der Javascript-Bereich der Website. Die Funktion setInterval() sorgt dafür, dass die Funktion getData() jede halbe Sekunde (500 Millisekunden) aufgerufen wird. Die Funktion getData() kümmert sich um folgenden darum Werte vom Arduino auf der Website

darzustellen. Sie verwendet dafür die sogenannte AJAX-Technologie, die verhindert, dass die Seite alle 500ms neu geladen werden muss und nur der relevante Teil der Seite, nämlich der neue Wert nachgeladen wird.

```
function getData()
            {
                var xhttp = new XMLHttpRequest();
                xhttp.onreadystatechange = function() {
                    if(this.readyState == 4 && this.status==200)
                    {
                        var x =
this.responseXML.getElementById("wert1").childNodes[0].nodeValue;
                        document.getElementById("wert").innerHTML = x;
                    }
                };
                xhttp.open("GET", "readValue", true);
                xhttp.send();
            }
```

Der relevante Teil der Funktion getData() ist: `var x = this.responseXML.getElementById("wert1").childNodes[0].nodeValue;` Hier wird eine XML-Datei, die der Webserver ausliefert ausgewertet. Vom Element mit dem Namen `"wert1"` wird der Inhalt in der Variablen x gespeichert.
Der Befehl `document.getElementById("wert").innerHTML = x;` kümmert sich darum, dass der Wert der Variablen x auf der Website im Bereich mit der ID *wert*, also `<span id="wert"></span>` angezeigt wird.
Der Befehl `xhttp.open("GET", "readValue", true);` kümmert sich darum, dass beim Webserver die XML-Datei readValue abgefragt wird.

Arduino-Code für den ESP8266

```
const char *ssid = "MINTCAMP01";
const char *password = "";

ESP8266WebServer server(80);
String html = "";
String rec = "test";
```

Es wird die SSID des WLAN und das Passwort festgelegt. Da das Passwort nicht gesetzt ist, ist das WLAN offen. Der Webserver wird mit Port 80 erzeugt. String html ist eine Variable in der später der Text der Website gespeichert wird. String rec ist die Variable, die die Werte des Arduinos speichert.

```
void setup() {
  delay(1000); //hier wird 1 Sekunde gewartet
  Serial.begin(115200); //Beginn der seriellen Kommunikation

  WiFi.softAP(ssid, password); // Start des Accesspoints

  IPAddress myIP = WiFi.softAPIP();// IP-Adresse des Webservers in myIP

  server.on("/", handleRoot);
  server.on("/readValue", handleValue);
  server.begin(); //Start des Servers
```

192.168.4.1/ aufgerufen, so kümmert sich die Methode handleRoot() darum, wird die URL 192.168.4.1/readValue aufgerufen, so kümmert sich die Methode handleValue darum. Die letztere Seite ist die XML-Datei, die von Javascript auf der Website angefordert wird.

```
bool ok = SPIFFS.begin(); //laden der Seite index.html vorbereiten

  if(SPIFFS.exists("/index.html")) //prüfen ob die Seite /index.html da ist
  {
      File f = SPIFFS.open("/index.html", "r"); //Öffnen der Datei zum lesen
      if(!f)
      {
        Serial.println("Something went wrong trying to open file...");
      }
      else
      {
        int s = f.size();
        Serial.printf("Size=%d\r\n", s);
        html = f.readString();
        f.close();//schließen der Datei
      }
  }
```

Der Befehl `html = f.readString();` kümmert sich darum, dass der Quelltext der Datei index.html, der oben beschrieben wurde in der Variablen html als Text abgespeichert wird.

```
void loop() {
  server.handleClient();

  if(Serial.available())
  {
    int x = Serial.read();
    rec = String(x);
  }
}
```

In der loop-Methode wird überprüft ob ein Client eine Seite ausgeliefert haben möchte. Sollte die Seite 192.168.4.1/ abgefragt werden, so kümmert sich die Methode handleRoot() darum, wird die 192.168.4.1/readValue abgefragt, so kümmert sich die Methode handleValue() darum.
Der Befehl `if(Serial.available())` prüft ob der Arduino Werte an den ESP übertragen möchte. Sollte das der Fall sein, so wird der Wert gelesen und in der Variablen x gespeichert. Diese wird in einen Text (String) umgewandelt und in der Variablen rec abgespeichert um später auf der Website angezeigt zu werden.

```
void handleRoot()
{
  int arguments = server.args();//Anzahl der GET-Parameter abfragen
  server.send(200, "text/html", html);
```

Die Methode handleRoot() wird aufgerufen, wenn ein Client die Website anfragt. Es wird überprüft ob, und wenn ja, wie viele GET-Parameter in der Seitenabfrage enthalten sind. Mit server.send() wird die Website zum Anzeigen ausgeliefert. Anschließend kümmert sich die Methode um die Auswertung der GET-Parameter.

```
String msg = "";
  for(int i=0; i<arguments; i++)
  {
    if(server.argName(i)=="power")
    {
      msg = msg + "<power>";

      if(server.arg(i)=="on")
      {
          msg = msg + "on";
      }
      if(server.arg(i)=="off")
      {
          msg = msg + "off";
      }
    }
    msg = msg + "<";
    byte buf[12];
    msg.getBytes(buf,12);
    Serial.write(buf,12);
    Serial.flush();
```

In der Variablen msg wird ein zunächst leerer Text gespeichert. Eine Schleife läuft über alle GET-Parameter, die in der Abfrage enthalten waren. Hier wird überprüft ob einer der Parameter den Namen „power“ hatte. Dieser Parameter wurde in der Website mit dem Link `href="/?power=on"` oder `href="/?power=off"` übergeben. Falls der Parameter in der URL enthalten war wird in der Variablen msg der Text <power> angehängt und überprüft, welchen Wert der Parameter hat. Für „on“ wird der Text „on“ an die Variable msg angehängt, für „off“ analog der Text „off“. Die Auswertung der Parameter kann für den Fall, dass weitere Parameter in der Website erstellt wurden oder weitere Werte übergeben werden, angepasst werden. Der Text in der Variablen msg wird mit einem `"<"` abgeschlossen. Über die Zeichen > und < kann später herausgefunden werden, wo sich die Werte im Text befinden.
Intern wird der so gestaltete Text der Variablen msg in ein byteArray mit der Länge 12 umgewandelt. Sollte die Nachricht mehr als 12 Zeichen enthalten muss hier eine neue Länge angegeben werden. Der Text wird dann als byteArray mit Serial.write() an den Arduino geschickt. Serial.flush() kümmert sich darum, dass der Sendevorgang sicher abgeschlossen ist.

```
void handleValue()
{
  String text = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>";
  text = text + "<node id=\"wert1\">";
  text = text + rec;
  text = text + "</node>";
  server.send(200, "text/xml", text);
}
```

Die Methode handleValue() wird von der Website im Javascript-Teil jede halbe Sekunde abgefragt. Sie erstellt einen Text der eine XML-Struktur enthält. `text = text + "<node id=\"wert1\">";` erstellt einen Knoten mit der ID *wert1*. Dieser Knoten wird später in der Webseite von Javascript gesucht (`this.responseXML.getElementById("wert1").childNodes[0].nodeValue;`)

In den Knoten wird jetzt der Wert aus der Variablen rec, der vom Arduino an den ESP geliefert wurde geschrieben und die Knotenstruktur beendet. Anschließend wird der XML-Text mit server.send() ausgeliefert. Sollten noch mehr Werte vom Arduino auf der Seite angezeigt werden, so können in diesen XML-Text noch weitere Knoten mit eindeutigen IDs eingefügt werden.

Arduino-Code für den Arduino Mega

```
int count = 0;
void setup() {
 Serial.begin(115200);
 Serial1.begin(115200);
}
```

Die Variable count zählt wie oft die Methode loop() bereits ausgeführt wurde. In der setup() Methode wird eine serielle Kommunikation mit dem Rechner über Serial.begin(115200) und eine mit dem ESP über Serial1.begin(115200) aufgebaut.

```
void loop() {
  count=count+1;
  if(Serial1.available())
  {
   readData();
  }
  if(count>100)
  {
    writeData();
    count = 0;
  }
}
```

In der Methode loop() wird die count-Variable bei jedem Ausführen um 1 erhöht. Sollte der ESP bereit zum Senden sein, so wird die Methode readData() aufgerufen. Die Methode writeData() zum Übertragen der Werte vom Arduino zum ESP soll nur jedes 100. Mal aufgerufen werden.

```
void readData()
{
    char buf[12];
    Serial1.readBytes(buf,12);
    String msg = buf;
    // zerlege den String z.B.: "<power>off<"
    int powerStart = msg.indexOf('>')+1;
    int powerEnd = msg.indexOf('<', powerStart);
    String power = "";

    for(int i = powerStart; i < powerEnd; i++)
    {
      power = power + msg.charAt(i);
    }

    Serial.println(power);
}
```

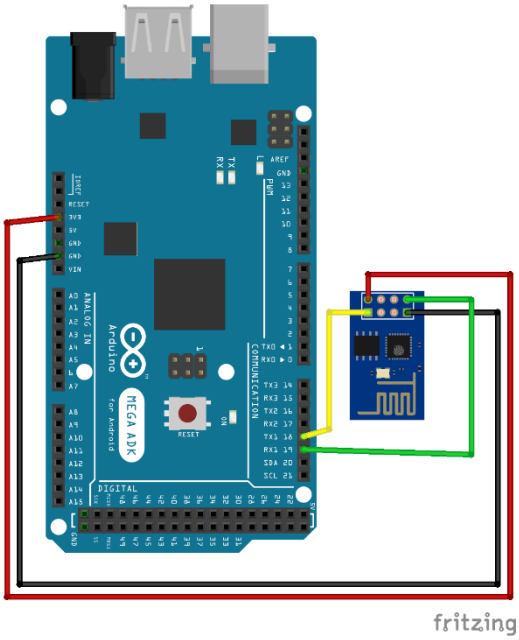
Die Methode readData() wertet den Text, den der ESP geschickt hat aus. Dafür wird die Nachricht in der Variablen buf (ein Feld mit Platz für 12 Zeichen) gespeichert und anschließend als Text zur Weiterverarbeitung in der Variablen msg abgelegt. Die Analyse der Nachricht orientiert sich an deren Aufbau. Es wird die Stelle im Text gesucht mit dem ersten Vorkommen des Zeichens `'>'`. Diese Zahl wird um 1 erhöht, da an dieser Stelle der Wert für die ursprüngliche Variable „power“ beginnt. In powerEnd wird das erste Vorkommen des Zeichens `'<'` nach der Position powerStart gesucht. An dieser Stelle endet der Wert für die ursprüngliche Variable “power“. In der for-Schleife wird dann der Wert Zeichen für Zeichen zusammengesetzt. Mit dem Befehl `Serial.println(power);` wird dieser Wert an den Computer geschickt. Im produktiven Einsatz würde anstatt des Sendens an den Computer der Wert ausgewertet (z.B. if(power==“on“){…}) und z.B. die Motoren gestartet.

Umgang mit dem ESP8266 und der Arduino-Software

Um Programme auf den ESP8266 zu laden muss dieser über das USB-Board an den Computer angeschlossen werden. Der Schalter des Boards muss dabei auf PROG stehen. In der Software muss das Board Generic ESP8266 Module ausgewählt werden. Als Flash-Size sollte 1M (512K SPIFF) ausgewählt werden. Der passende COM-Port muss ausgewählt werden (notfalls im Geräte-Manager des Betriebssystems nachlesen). Das Programm kann dann an den ESP wie beim Arduino übertragen werden.
ACHTUNG: Nach jedem Hochladen des Programms muss der ESP bzw. das USB-Board einmal vom USB-Port abgezogen werden um das Programm erneut aufzuspielen. Es ist normal, wenn das Übersetzten und das Hochladen länger dauert als beim Arduino.

Um die Datei index.html auf dem ESP abzuspeichern muss diese sich in dem „Sketchverzeichnis“, des Programms das auf die Datei zugreift in einem Ordner mit Namen data liegen. Über Werkzeuge -> ESP8266 Sketch Data Upload wird die Datei auf den ESP geladen und dort abgespeichert. Auch dieser Vorgang kann etwas Zeit in Anspruch nehmen. Danach muss ebenfalls die USB-Verbindung kurzzeitig getrennt werden um erneut Programme oder Dateien auf den ESP zu laden.

Anschluss des ESP8266 an den Arduino



Einführungsworkshop Arduino

Das EVA-Prinzip beim Arduino

Sensoren (Eingabe)

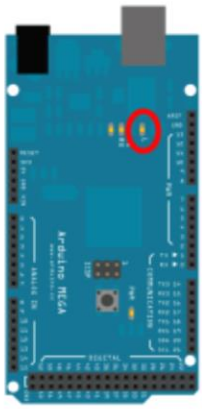
Microcontroller (Verarbeitung)

Aktoren (Ausgabe)

Den Arduino Programmieren

- Die Arduino IDE als zentrale Entwicklungsumgebung
- Programme enthalten immer die Methode void setup(), die einmalig aufgeführt wird
- Programme enthalten immer die Methode void loop(), die dauerhaft wiederholt ausgeführt wird.
- Programme werden übersetzt und auf den Arduino geladen

Bringe die interne LED zum blinken



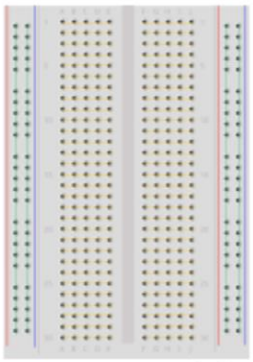
Die eingebaute LED ist an PIN 13 angeschlossen

Programmierung:

1. PIN als Ausgang definieren:
   `pinMode(13, OUTPUT);`
2. LED einschalten:
   `digitalWrite(13, HIGH);`
3. Warten:
   `delay(1000);`
4. LED ausschalten:
   `digitalWrite(13, LOW);`

Schaltungen aufbauen

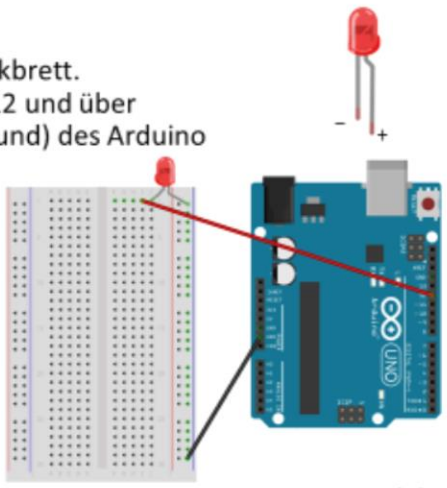
- Um weitere Aktoren und Sensoren an den Arduino anzuschließen, verwendet man ein sog. Steckbrett.
- In das Steckbrett können Bauteile und Kabel einfach eingesteckt werden.
- Die Löcher auf dem Steckbrett sind untereinander folgendermaßen verbunden:
  - Links und Rechts außen sind die Spalten durchgehende verbunden
  - Im inneren Bereich sind jeweils die „Zeilen“ links bzw. rechts untereinander verbunden.



fritzing

Externe LED blinken lassen

- Steckt eine rote LED wie abgebildet auf euer Steckbrett. Über das rote Kabel ist sie mit dem digitalen Pin 12 und über das schwarze Kabel mit dem Minuspol (GND, Ground) des Arduino verbunden.
- Das kürzere Beinchen der LED wird immer näher zum **Ground** (Minus-Pol) ausgerichtet.
- Lasst jetzt die rote LED blinken.

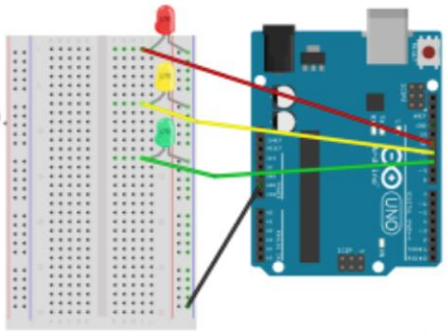


fritzing

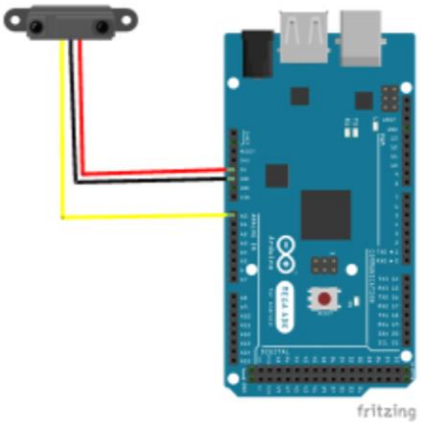
Ampelschaltung

- Ergänzt die Schaltung um eine gelbe und eine grüne LED.
  Achtet wieder darauf, dass das kürzere Beinchen der LED mit **Ground (GND)** verbunden ist.

- Programmiert den Ablauf einer Ampelschaltung(rot, rot-gelb, grün, gelb, ...).



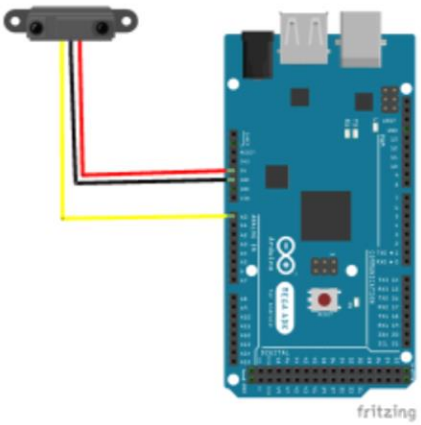
Analoge Sensoren



Lese Entfernungswerte mit dem Infrarotsensor und schicke diese zum PC:

1. Starten des Seriellen Ports:
   `Serial.begin(9600);`
2. Speichern des Spannungswertes an A0 in einer Variablen:
   `int val = analogRead(A0);`
3. Ausgabe des Wertes:
   `Serial.println(val);`

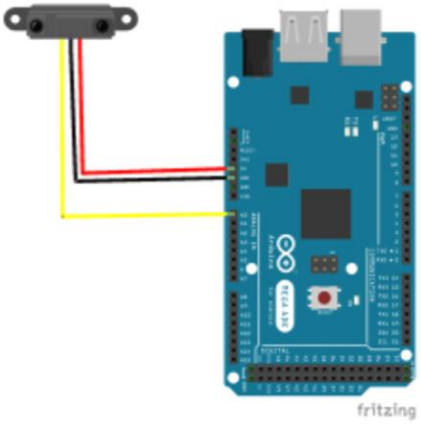
Analoge Sensoren



Berechnen korrekter Entfernungswerte:

- Recherchiere im Internet, mit welcher Formel die Daten in korrekte Entfernungen umgewandelt werden können.
- Achte darauf, welche Version des Sensors du verwendest
- Gib die berechneten Entfernungswerte in cm aus.

Analoge Sensoren



Formulas

These formulas are derived from the Sharp datasheets to compute distance.
The formula to translate SensorValue into Distance for Sharp 10-80cm analog sensors is:

Distance (cm) = 4800/(SensorValue - 20)

This formula is only valid over the SensorValue range 80-500.

The formula to translate SensorValue into Distance for Sharp 20-150cm analog sensors is:

Distance (cm) = 9462/(SensorValue - 16.92)

This formula is only valid over the SensorValue range 80-490.

The formula to translate SensorValue into Distance for Sharp 4-30cm analog sensor is:

Distance (cm) = 2076/(SensorValue - 11)

This formula is only valid over the SensorValue range 80-530.

For digital distance sensors, SensorValue will be greater than 200 if the distance of the object being measured is less that the detection distance of the sensor. Otherwise the SensorValue will be less that 200.

Digitale Sensoren

Überprüfe ob sich ein Objekt im Bereich des Lasers befindet (ca. 80cm)

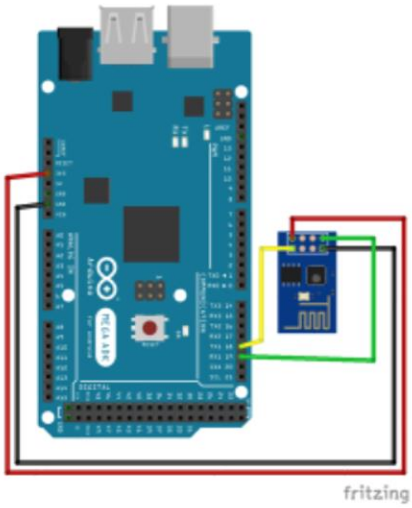
1. Starten des Seriellen Ports
2. Auslesen der Information: digitalRead(Pin)
3. Ausgeben der Information am Seriellen Monitor

Erweiterung: Eine LED soll solange leuchten wie sich ein Objekt im Bereich des Lasers befindet.

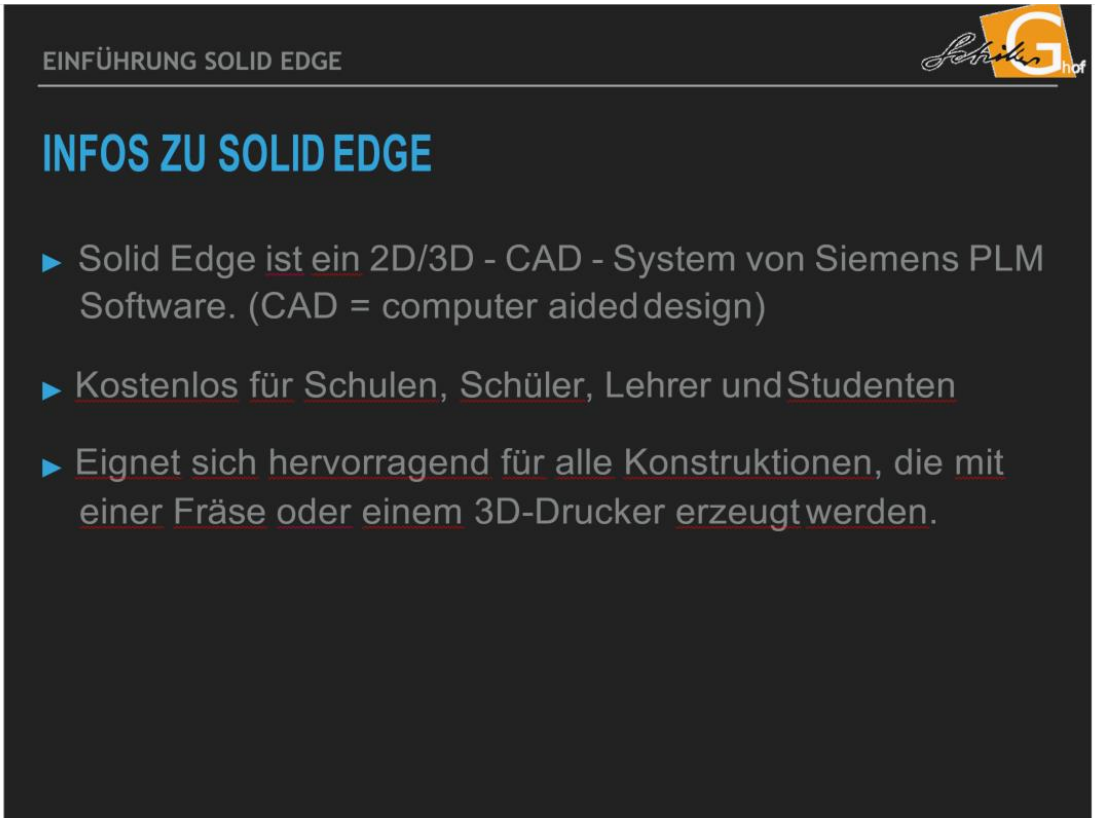
WiFi-Shield

Empfangen von Daten die über das WLAN des WiFi-Shields empfangen und an den Arduino geschickt werden.

1. Verbinden des Shields mit dem Arduino (ACHTUNG 3.3V anstatt 5V nutzen)
2. Mit dem Smartphone passendes WLAN auswählen (SSID: MINTCAMP0X)
3. URL 192.168.4.1 im Browser des Smartphones aufrufen
4. Programm WIFI_Mega auf den Arduino übertragen und Seriellen Monitor aufrufen
5. Betätigen des Links (on – off) am Smartphone




WORKSHOP 3D-MODELLIERUNG
SOLID EDGE


EINFÜHRUNG SOLID EDGE
INFOS ZU SOLID EDGE
Solid Edge ist ein 2D/3D - CAD - System von Siemens PLM Software. (CAD = computer aided design)
Kostenlos für Schulen, Schüler, Lehrer und Studenten
Eignet sich hervorragend für alle Konstruktionen, die mit einer Fräse oder einem 3D-Drucker erzeugt werden.

GENERELLER WORKFLOW

- Erstellung von 3D-Modellen mit Solid Edge (z.B. Halterungen für Sensoren)
- Export des Modells als .STL-Datei
- Öffnen der .STL-Datei in der Software Cura (Ultimaker)
- Anpassung der Einstellungen und Export des Schichtenmodells auf eine SD-Karte
- Drucken des Modells mit Hilfe eines 3D-Druckers

SOLID EDGE (ORDNER MINT CAMP AUF DEM DESKTOP)

- Öffne Solid Edge
- Erstelle über NEU ein neues iso-metrisches Teil
- WICHTIG: Ändere den Synchronous-Mode per Rechtsklick auf Sequentiell ab (mit Historie)

STEUERUNG DER SOFTWARE

- Drücken des Mausrades ermöglicht 3D-Bewegungen im Raum
- Würfel (rechts unten) ermöglicht sofortigen Switch in bestimmte Ansichten
- Home-Button (neben Würfel) ermöglicht sofortigen Switch in Ausgangsansicht

ERSTE SKIZZE

- Klick auf Skizze (2D)
- Auswahl der Ebene für Skizze
- Skizzieren eines Grundrisses (geschlossene Linienfolge), z.B. Rechteck oder Kreis
- Skizze schließen (oben rechts)
- Bezeichnung von Skizze festlegen und Fertig anklicken

EXTRUSION

- Klick auf Extrusion (Dialogfenster öffnet sich)
- Im Dropdown-Menü —> Aus Skizze wählen
- Auswahl der soeben erzeugten Skizze
- Linksklick auf den grünen Haken
- Extrusion der Skizze (Linksklick legt Größe der dritten Dimension fest)

WEITERE MÖGLICHKEITEN

- Übergang (Pyramidenförmige Halterung)
- Ausschnitt
- Rotation (runde Halterung mit Loch)
- Bohren (Schraubenloch)
- Verrundung (schönere Optik und bessere Haptik)
- Formschräge
- etc.

Feedback der SuS zum MINT-EC-Camp, Hof, Juni 2018-07-09

a) Punktebewertung

(beste mögliche Bewertung = 5, schlechteste mögliche Bewertung = 1)

a. "Wie haben Dir die die Inhalte der Veranstaltung gefallen?" → 5,0

b. "Wie haben Dir das Niveau und die Art der Wissensvermittlung gefallen?" → 4,7

c. Unabhängige Gesamtbewertung → 5,0

b) Anmerkungen der SuS zur Frage…

a. "Welcher Teil der Veranstaltung hat Dir besonders gefallen und warum?"

i. 3D-Druck, weil das einfach faszinierend ist

ii. Das eigenverantwortliche Arbeiten

iii. Selbständiges Arbeiten, Betriebsbesichtigung, Vortrag von Professor. Ich fand die Mischung gut. Zudem hat alles gut zum Thema gepasst und war abwechslungsreich

iv. Das eigenständige Arbeiten: Es war nicht langweilig und wurde dem entsprechendem Niveau angepasst

v. Das eigenständige Arbeiten, weil man selbst alles austesten konnte und trotzdem immer Hilfe durch die Lehrer bekommen konnte.

vi. Das eigenständige Programmieren, weil man eigentlich alles selber machen konnte aber immer Hilfe bekam wenn man sie brauchte

b. "Was nimmst Du aus der Veranstaltung mit?"

i. Viele neue Erfahrungen und mehr Verständnis dafür, warum die Entwicklung so lange dauert.

ii. Dass vollständige autonome Autos noch in weiter Zukunft liegen

iii. Dass man bei Programmierarbeiten manchmal lange nach Fehlern suchen muss. Aber auch, dass es viel Spaß macht, neue Dinge zu entwickeln.

iv. Verbesserte Programmierkenntnisse

v. Neue Freunde, Kenntnisse zu dem Arduino

c) Gesamt-Kommentar (ohne vorangegangene Frage)

a. Ein sehr gelungenes Camp auf hohem Niveau

b. Super organisiert, motivierte und motivierende Lehrer

c. Ich fand es schade, dass es Teilnehmer gab, die keinen Wert auf Teamarbeit legten und die nur ihre eigenen Interessen verfolgten. Ansonsten fand ich es einen gelungenen und lehrreichen Workshop.

d. Die Veranstaltung hat mir sehr gut, aufgrund der vielen praktischen Einheiten, gefallen. Jedoch finde ich, dass die Vorträge entweder am Anfang oder am Ende sein sollten da man sonst aus einigen Ideen gerissen wird, um sich einen Vortrag anzuhören der zur jetzigen Zeit nicht relevant ist, sondern auch gut am Ende sein könnte.

e. Ich hatte viel Spaß das Modell erst einmal zusammenzubauen, dann zu programmieren und die Fehler zu beheben, die sich immer mal wieder eingeschlichen haben, da es nie große Lücken gab in denen wir nichts tun konnten. Außerdem fand ich es sehr gut das wir so viele verschiedene Bereiche hatten, die wir kennengelernt haben, wie: das CAD Programm, den Schaltplan, die Bauanleitung an sich und so weiter.

Bezugsquellen für Hardware

- Automodell Elecro Smart Car 4wd z.B.:
  - https://www.elecrow.com/4wd-smart-car-robot-chassis-for-arduino-servo-steering.html
  - https://www.roboter-bausatz.de/1659/4wd-rc-smart-car-chassis
- Akkupack (9,6 V NiMh 800mAh) z.B.:
  - https://www.conrad.de/de/conrad-energy-modellbau-akkupack-nimh-96-v-800-mah-zellen-zahl-8-block-tamiya-stecker-209081.html
- Motorsteuerung (L298N DualHBridge):
  - Diverse Elektronikshops
- Servo – Stromversorgung 5V-Step-Down Module (z.B. 5A XL4015 DC-DC Step Down Adjustable):
  - Diverse Elektronikshops
- Erweiterungsplatten zur Befestigung (Pololu RP5 Expansion Plate):
  - https://www.pololu.com/product/1531