HOCHSCHULE
HAMM-LIPPSTADT

Sommersemester 2022-
Wintersemester 2022/23
Praxissemester:
01.08.22 – 28.02.2023

Hochschule Hamm-Lippstadt
Praxissemester
Betreuer/in: Prof. Schneider

- Projektarbeit-

Einleitung

Der Schulweg von Kindern ist nicht sehr sicher. Durch Geschwindigkeitsmessungen des ADAC vor 25 Schulen in zehn Bundesländern wurden häufig Fahrer mit zu hohen Geschwindigkeiten gemessen. Die zulässige Höchstgeschwindigkeit von 30 km/h wurde von mehr als 60 Prozent der Autofahrer überschritten. Von insgesamt 43 828 Messungen, waren 26 329 zu schnell unterwegs. Vor einer Grundschule in Hamburg wurde ein Rekordwert von 96 km/h bei vorgeschriebenen 30km/h gemessen.

Um die Autofahrer abzuschrecken, können sogenannte Tempomessgeräte helfen.
Diese messen die Geschwindigkeit und zeigen diese über eine Anzeige dem vorbeifahrenden Fahrer. Zudem wird oft auch noch mit einem Smiley signalisiert, ob der Fahrer zu schnell fährt oder ob er sich an die vorgegebene Geschwindigkeit hält.

Aufgabenstellung

Die Aufgabe des Projektes war es, ein solches Tempomessgerät mit einem Radarsensor und einer LED-Anzeige, welche die Geschwindigkeit in km/h anzeigt, zu entwickeln.

Anforderungen

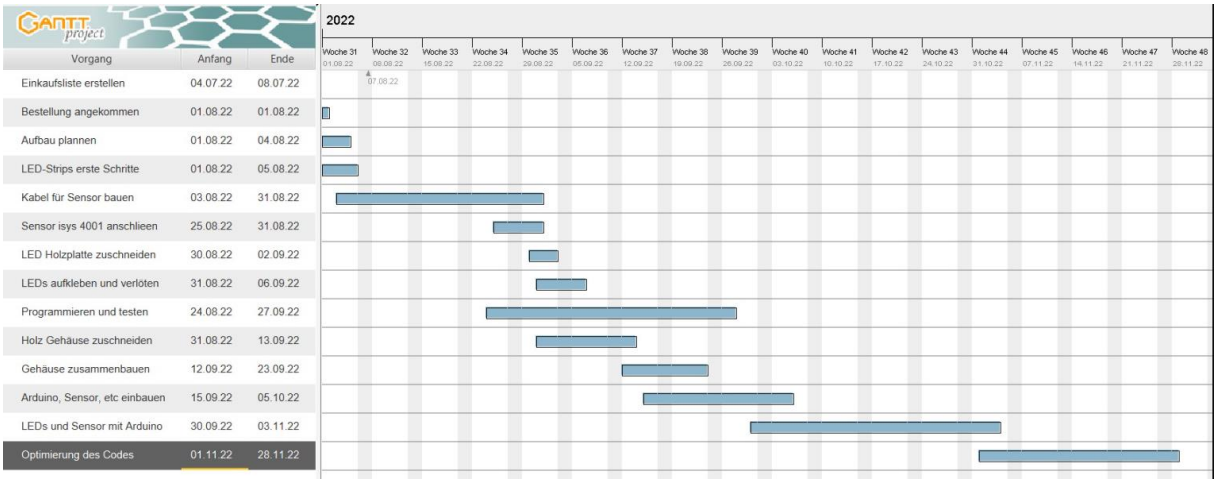
Tabelle 1: Lastenheft

#	Anforderung
Req 01	Das System muss die Geschwindigkeit bis 100 km/h messen und anzeigen.
Req 02	Nutzen Sie den RADAR-Sensor iSYS-4001 der Firma InnoSenT.
Req 03	Die Reichweite muss min. 50 m betragen.
Req 04	Das System muss hochleistungs-LED-Anzeige in gelb und rot verwenden und im hellen Tageslicht gut lesbar sein.
Req 05	Das System muss rot bei überhöhter Geschwindigkeit diese rot blinkend anzeigen.
Req 06	Das System soll die Geschwindigkeit mit einer Zahlenhöhe von 30 cm 2-stellig anzeigen.
Req 07	Das System muss eine Smiley-Funktion ☹ ☺ anzeigen (vgl. Abb. 1). Der Smiley muss ab einem einstellbaren Geschwindigkeitsschwellwert umgeschaltet werden.
Req 08	Die Anzeige soll im Hochformat dem Format (84x63x18) cm entsprechen.
Req 09	Das System soll wetterfest sein.
Req 10	Kür: Das System erstellt bei Geschwindigkeitsüberschreitung ein "Blitzerfoto" (vgl. Abb. 2), welches ein Beweisbild inkl. Maximalgeschwindigkeit, Datum und Uhrzeit speichert.

Tabelle 2: Pflichtenheft

#	Anforderung
nFA1	Die LEDs sollen über einen Microcontroller gesteuert werden.
nFA2	Die Geschwindigkeit soll mit C++ programmiert werden.
nFA3	Es muss ein Radarsensor verwendet werden.
nFA4	Die Geschwindigkeit soll nach 1 Sekunde angezeigt werden.
nFA5	Die Geschwindigkeit muss in km/h angezeigt werden.

Projektplan (Justin Frommberger und Jonas Gerken)



BOM (Justin Frommberger und Jonas Gerken)

Pos.	Anz.	Bezeichnung	Artikelnummer
1	1	Arduino UNO	Gegeben
2	1	5 Meter RGB LED-Steifen WS2812B(60LEDs/Meter)	LT5050RGBIC593605005IP20 bei ledtech-shop.de
3	1	Radarsensor IsYs 4001	2191-80.00000191-ND bei digykey.com
4	1	M12 8 poliger Stecker	123-1064 bei RS-online.de
5	1	RS232 Sub-D Stecker(zum Löten)	Gegeben
6	1	Netzteil 5V 4A	MW GST25E05 bei reichelt.de
7	2	2m Datenkabel	Gegeben
8	1	Wiederstand 220 Ohm	Gegeben
9	1	Plexiglasscheibe 500mm x 500mm x 2mm	AXT-020-TRAEVO050050 bei expresszuschnitt.de
10	2	MDF-Platten 1500mm x 1500mm x 11mm	Gegeben
11	1	InnoSenT Kabel für Sensor ISyS 4001	über Anfrage an InnoSenT erhalten

Funktionaler Systementwurf (Justin Frommberger)

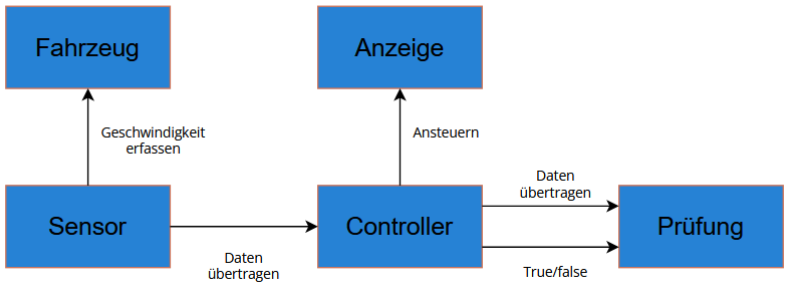


Abb. 1 Funktionaler Systementwurf

In Abb. 1 wird gezeigt, wie die Komponenten verbunden sind.
Der Sensor erfasst die Geschwindigkeit des Fahrzeugs und übergibt diese dem Microcontroller. Mit dem Mikrocontroller werden die Daten berechnet, die er vom Sensor empfängt. Die berechneten Werten werden dann auf der Segmentanzeige dargestellt.

Technischer Systementwurf (Jonas Gerken)

Abb. 2 Technischer Systementwurf

Die Abbildung 2 zeigt den technischen Systementwurf des Tempomessgerätes als CAD-Modell.

Radarsensor Isys 4001 (Jonas Gerken)

Der Kabelanschluss ist in Abb. 3 grafisch dargestellt.

Dargestellt ist der Female 8-polige Stecker, der mit dem Geschwindigkeitssensor verbunden wird. Das andere Ende des Kabels wird aufgeteilt in eine RS232-Schnittstelle und einem Stecker zur Verbindung mit dem Mikrocontroller der auch die Spannungsversorgung für den Sensor bereitstellt. Die RS232-Schnittstelle wird mit dem Computer verbunden zur Übertragung der Daten.

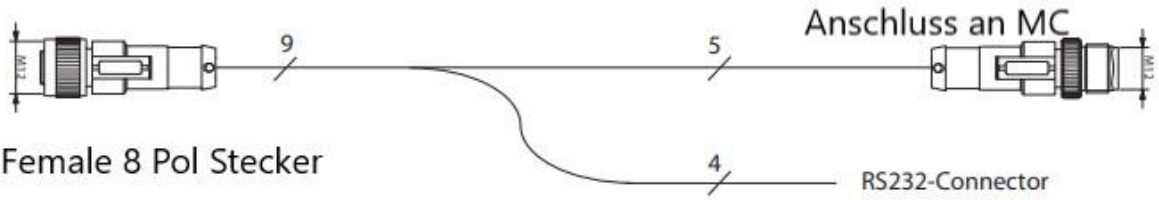


Abb. 3: Sensoranschluss

Um das Kabel anzuschließen, werden zwei Datenkabel mit Abschirmung benötigt. Das Kabel muss eine Abschirmung haben, um Störsignale zu vermeiden.

Für den RS232-Connector muss ein Kabel an einen Sub-D Stecke mit lötkontakten gelötet werden (Tabelle 3).

Wichtig ist es, die einzelnen Adern mit einem kleinen Schrumpfschlauch zu versehen, damit die Aderisolierung beim Löten nicht beschädigt wird. Das zweite Kabel ist für die Ausgänge Out1, Out2, Out3, genauso wie die Stromversorgung und der Ground.

Tabelle 3

RS232	Signal	Aderfarbe	8 Pol female
Pin 2	RX(Recieve)	Orange	Pin 7
Pin 3	TX(Transmit)	Grün	Pin 8
Pin 4	not connected	blau	Pin 4
Pin 5	Ground	Alle weißfarbigen Adern orangeweiß etc.	Pin 5

Für das zweite Kabel, welches für die Datenoutputs und die Stromversorgung verwendet wird, sieht die Pin-Belegungen wie folgt aus. (Tabelle 4)

Tabelle 4

8 Pol female	Signal	Aderfarbe	Microcontroller?
Pin 1	Out1	Blau	Digital Pin
Pin 2	Out2	Grün	Digital Pin
Pin 3	Out3	orange	Digital Pin
Pin 5	VCC	braun	Powersupply
Pin 6	Ground	weiß + Farbe	Ground

Da der Sensor mit dem selbst gebauten Kabel nach mehreren Überprüfungen der Verbindungen nicht mit dem GUI-Programm verbunden werden konnte, wurde ein Kabel von der Firma InnoSenT angefragt. Mit diesem Kabel war es dann möglich, den Sensor mit dem GUI zu verbinden und zu konfigurieren.

Sensormessung

Damit ein PWM-Signal vom Sensor gesendet wird, muss zunächst ein Pull-up Widerstand (10kOHM) wie in Abb. 4 gezeigt, zwischen den Outputs geschaltet werden. Bei dem Anschluss an den Arduino muss beachtet werden, dass die Ausgänge OUT 1-3 an die PWM-Schnittstellen angeschlossen werden.

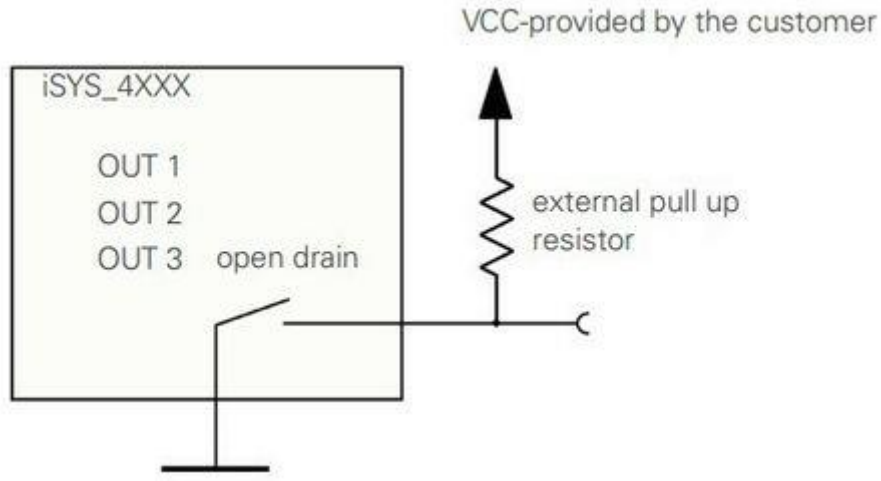


Abb. 4 OUT-Anschlüsse/Verkabelung

Das Signal kann nach der Verkabelung über ein Oszilloskop angezeigt werden. (Abb. 5)
Das obere PWM-Signal ist die Entfernung und das untere die Geschwindigkeit.

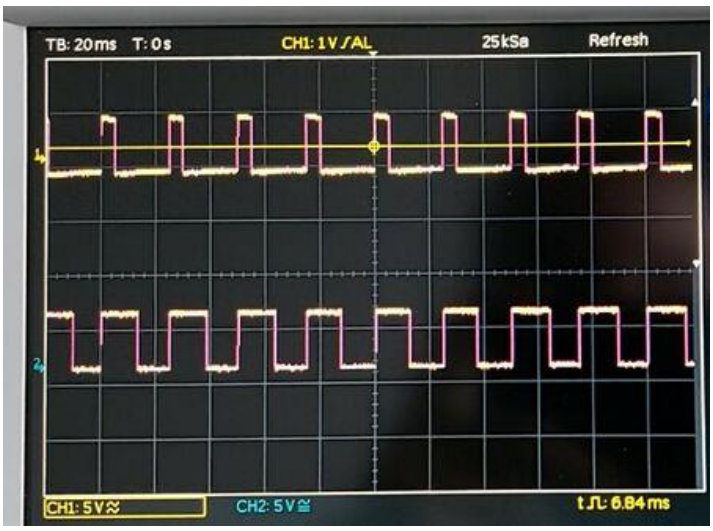


Abb. 5 PWM-Signal

Der Sensor sendet die PWM-Werte an den Arduino. Mit dem Arduino Befehl pulseIn(pin, HIGH), kann das Signal ausgelesen werden.

Mit den folgenden Formeln (Abb. 6) wird aus einer abgelesenen Geschwindigkeit das DC und folgend das „Ton“ berechnet.
Der Sensor sendet Ton-Werte an den Arduino und dieser muss in eine Geschwindigkeit umgerechnet werden.
Dazu muss die obere Formel nach km/h umgestellt werden und die untere Formel nach DC.
Rechts sind die umgestellten Formeln aufgezeigt, um die Geschwindigkeit (v) zu berechnen.

$$DC = \frac{25.1km/h - 0km/h}{100km/h - 0km/h} * 100 = 25.1\%$$

$$T_{ON} = 25.1\% * 25ms = 6.275ms$$

$$DC = \frac{6{,}275}{25} \cdot 100$$

$$v = \frac{25 * 100}{100}$$

Abb. 6 Duty Cycle Formel

LED-Seven-Segment Aufbau und Verlötung (Justin Frommberger)

Der Erste Schritt ist, für die LED-Seven-Segmente eine Holzplatte auf 500 mm x 500 mm zuzuschneiden.
Auf dieser Holzplatte werden dann die LED-Stripes aufgeklebt.

Wir haben uns für die Seven-Segmente anzeige entschieden.

Diese LED-Strips haben wir doppelt nebeneinander geklebt, damit die angezeigte Geschwindigkeit später auf der Straße besser zu lesen ist.

Dazu wurden 8 Strips von einer Länge von 20 LEDs und 12 Strips von einer Länge von 8 LEDs zugeschnitten und angeklebt.

Abb. 7 LEDs auf Holzplatte

Beim Aufkleben ist es wichtig, die LEDs richtig herum aufzukleben.

Drauf geachtet werden muss, dass man mit der Kennzeichnung Strips "Di" für Digital In beginnen muss.

Der erste Strip wurde von oben nach unten geklebt, der Stripe rechts daneben von unten nach oben um die Reihenfolge einzuhalten.

So müssen unten später nur 3 kurze Kabel angelötet werden.

Von links außen und rechts außen wurden 50 mm und von dem unteren Rand wurden 15 mm Platz bis zu den LED-Stripes gelassen.

Abb. 7 zeigt die fertige Segmentanzeige.

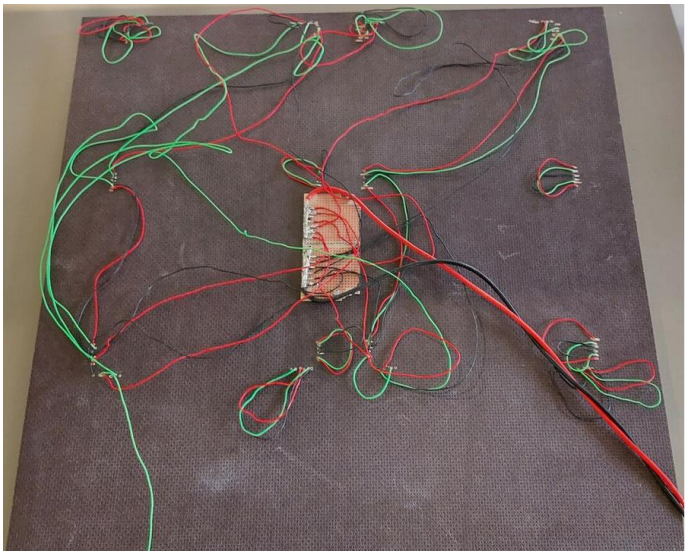


Abb. 8: Verkabelung

Bei einer so großen Anzahl an LEDs ist es notwendig, die LEDs an mehreren Stellen mit der Stromquelle zu verbinden.

Dazu wurden, wie in Abb. 8 zusehen ist ein rotes und ein schwarzes 4 mm² Kabel an eine Europlatine gelötet, welche dann die Hauptstromversorgung darstellt.

Die Seven-Segmentanzeige wurde dafür jeweils in drei Abschnitte unterteilt.
Das Seven Segment wurde wie folgt aufgeteilt:
- die linken senkrechten LEDs
- die rechten senkrechten LEDs
- die waagerechten LEDs

Auf der Holzplatte wurden an jeder Ecke 3 Löcher gebohrt, um die Kabel zur Rückseite durchzuführen und dort zu verdrahten.

Für die Verbindungen zwischen den Strips, wurde ein 0,5 mm² Kabel in zwei verschiedenen Farben: Rot (Strom), Schwarz (Erde), Grün (Daten) zugeschnitten und verlötet.

Das Datenkabel wird nicht in Reihen aufgeteilt, denn alle LEDs müssen zur Programmierung in Reihe angeschlossen werden. So kann später mit Variablen jede einzelne LED angesteuert werden.

Die Abschnitte werden jeweils an die Europlatine angelötet, wie auf dem Bild zu sehen ist. (Abb. 8)

LED Dimmung (Jonas Gerken)

Um die Autofahrer mit den LED‘s nicht zu blenden, haben wir uns überlegt, eine Abdeckung mit dem 3D-Drucker zu entwerfen um das Licht zu dimmen. Die einzelnen Bauteile für die Abdeckung wurden mit SolidWorks gezeichnet und in eine STL Datei umgewandelt. Die Einzelteile die mit dem 3D-Drucker gedruckt wurden, haben wir anschließend auf die Holzplatte über die LED‘s festgeklebt.

In Abb. 9 ist dann das Ergebnis der fertigen Abdeckung zusehen.

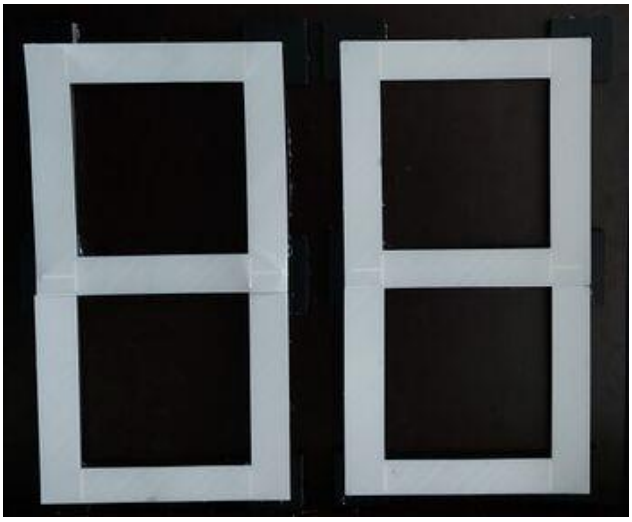


Abb. 9 LED-Abdeckung

Gehäuse (Justin Frommberger)

Für das Gehäuse wurden drei 11 mm dicke Holzplatten auf 250 mm * 500 mm zugeschnitten. Eine von den Holzplatten wird mit drei Nuten versehen, die Nuten sind zur befestigen der LED-Platte, der Plexiglasscheibe und der Rückwand.

Mit der zweiten und dritten Platte wird die Außenwand befestigt. Die Platte für die Rückwand wird aus einer 11 mm dicken Holzplatte auf 500 mm * 500 mm zugeschnitten. Zudem muss auch die Plexiglasscheibe auf 500 mm * 500 mm zugeschnitten werden, um in die dafür vorgesehene Halterung zu passen. Am Ende schnitten wir die letzte Platte auf 500 mm * 500 mm und klebten auf diese Platte, die 7 Segment LED Pixelstreifen.

Wir montierten alle Teile des Gehäuses zusammen und befestigten diese zusätzlich mit Schrauben für einen besseren Halt.

Der fertige Aufbau ist in Abb. 10 zusehen.

Abb. 10 Gehäuse

Elektronik (Jonas Gerken)

Die Abb. 11 zeigt die Elektrische Schaltung.

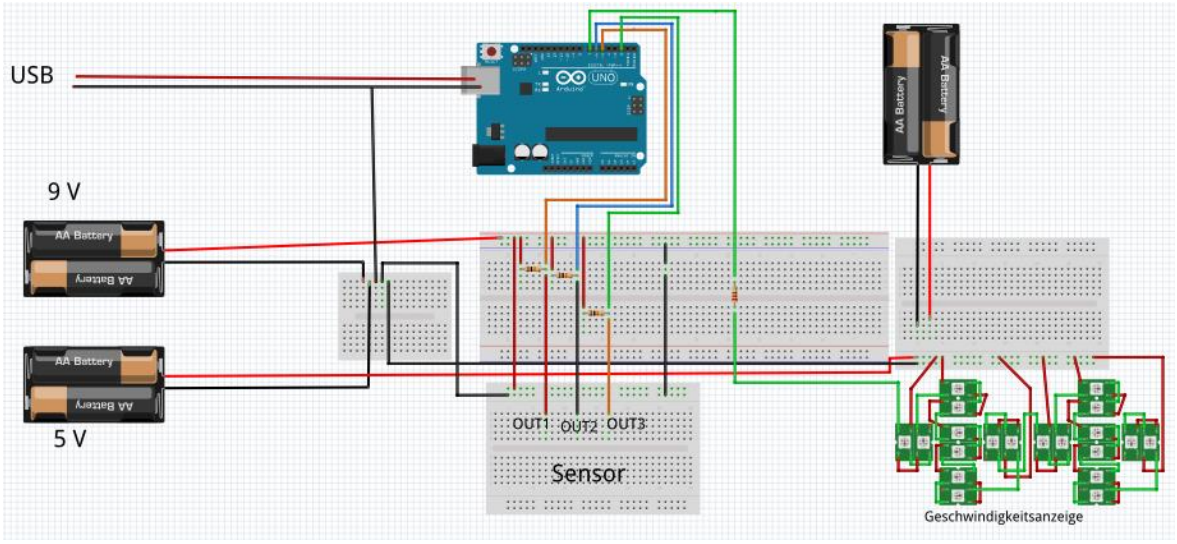


Abb. 11: Schaltung

Stromquellen:

- Radarsensor: 5 bis 30 Volt (in Abb. 7 links)
- LED-Geschwindigkeitsanzeige: 5 Volt (in Abb. 7 rechts)

Arduino Uno:

- Digital Pin 3, 5, 6: OUT1, OUT2, OUT3 des Radarsensors
- Digital Pin 7: Digital der Geschwindigkeitsanzeige

Radarsensor:

- OUT1, OUT2, und OUT3 mit einem PULL-UP Widerstand verbunden

LED-Geschwindigkeitsanzeige:

- Senkrechten LED-Streifen jeweils links und rechts mit der Stromquelle verbunden
- die mittleren LED-Streifen jeweils links und rechts mit der Stromquelle verbunden

Programmierung

Farbe der LEDs (Justin Frommberger)

```
void switchCase() {
  int LINKS = 0;
  int RECHTS = 0;
  int Rot = 0;     //Deklaration um LEDs Rot oder Gruen anzuzeigen
  int Gruen = 0;

  LINKS = round(v) / 10;   // linke Ziffer herausfinden
  RECHTS = round(v) % 10;  // rechte Ziffer herausfinden

  for (int i = 0; i <= 255; i++) {
    leds[i] = CRGB(0, 0, 0);
  }

  if (LINKS <= 3 && RECHTS <= 9) {   // Geschwindigkeiten unter 30 KM/H werden Gruen angezeigt
    Rot = 0;
    Gruen = 150;
  }
  if (LINKS >= 3 && RECHTS >= 0) {   // Geschwindigkeiten über 30 KM/H werden Rot angezeigt
    Rot = 150;
    Gruen = 0;
  }
```

Abb. 12 LED-Programmierung (Farbe)

Zahlen Anzeige (Justin Frommberger)

```
switch (LINKS) {
 case 0:
    for(int i=0; i<=55; i++)
    {
    leds[i] = CRGB(Rot, Gruen, 0);
    }
    for(int i=72; i<=125; i++)
    {
    leds[i] = CRGB(Rot, Gruen, 0);
    }
     break;
  case 1:
    for(int i=88; i<=125; i++)
    {
    leds[i] = CRGB(Rot, Gruen, 0);
    }
     break;
```

Abb. 13 Zahlen Anzeige

Um die Zahlen auf dem Display anzuzeigen, wurde ein Switch Case programmiert, um die benötigten LEDs einzuschalten. Wie im Beispiel Case 0, werden alle LEDS von 0 bis 55 und von 72 bis 125 eingeschaltet.

Sensor (Jonas Gerken)

```
  for (int i = 0; i < 5; i++) {       // 5 Messungen durchführen
    ontime = pulseIn(pin, HIGH);      // Messung wie lange der Sensor ein High Signal sendet

    speedarray[i] = ontime;           // Messungen in Array speichern
    delay(250);
  }

  sortArray(speedarray, 5);          // Array Inhalt nach Größe sortieren
  mitte = speedarray[2];             // Median-Wert herausfinden

  if(mitte > 0 && mitte <= 25000){   // nur postive ontimes
  v = (mitte / T) * 100;
  }
  else {
  v = 0.0;
  }
```

Abb. 14 Sensor Programmierung

Ergebnis (Justin Frommberger und Jonas Gerken)

Fast alle Anforderungen des Projektes konnten erfüllt werden. Wir unterschätzten, dass die LED‘s eine Menge an Strom benötigen (aktueller Stand 10A), so mussten wir bei unserem Projekt LED‘s einsparen und beschlossen das Smiley Gesicht nicht einzubauen.
Um trotzdem die gewünschte Aufgabe zu erfüllen, überlegten wir uns, dass wir die nicht überschrittene Geschwindigkeit Grün anzeigen lassen.
Die zu hohe Geschwindigkeit stellen wir in roter Farbe dar. Zudem haben wir uns für ein kleineres Format des Schildes entschieden, um unser Projekt besser transportieren zu können. Auch haben wir uns gegen einen Schriftzug vom km/h entschieden, um die Zahlen möglichst groß und sichtbar für die Fahrer zu gestalten. Leider haben wir festgestellt, dass der vorhandene Radarsensor Isys 4001 eine Störung hat und keine konstanten Werte zurückliefert, dieses führt bei der Anzeige zu ungenauen Werten.
Alle anderen oben aufgelisteten Anforderungen konnten wir erfüllen.

Zusammenfassung (Justin Frommberger und Jonas Gerken)

Lessons learned

Das Team hat die folgenden Kompetenzen erlernt:

- Projektplanung und Zeitmanagement
- CAD-Konstruktion für 3D Druck
- 3D Druck
- Löten
- Ansteuerung von WS2812 LEDs
- Bau eines Kabels

Besondere Herausforderungen

Die erste Herausforderung bei unserem Projekt „Speed-Tempomessgerät“, stellte sich direkt am Anfang bei der Materialbeschaffung. Das Problem war es, das alle Materialien, die für das Projekt benötigt werden zu beschaffen. Wir stellten fest, dass bei dem vorhandenen Radarsensor der Hochschule das Kabel zur Strom- und Datenversorgung fehlte. Weil das Kabel kein Standardkabel war, informierten wir uns bei dem Hersteller des Sensors wie das Kabel aufgebaut ist und ließen uns das dazugehörige Datenblatt schicken. Um das Kabel selbst anzufertigen, bestellten wir dafür einen RS232 Sub-D Stecker und einen M12 8 poligen Stecker. Das Herstellen des Kabels konnte nach langem ausprobieren und telefonischem Austausch mit der Herstellerfirma nicht abgeschlossen werden. Folgend fragten wir die

Firma, ob sie uns ein Kabel zuschicken könnten. Freundlicherweise ließen sie uns ein Kabel kostenlos zukommen, um unser Problem zu lösen.

Die nächste Herausforderung war es, herauszufinden, wie groß die Netzteile seien müssen, um unser komplettes Projekt mit Strom zu versorgen. So recherchierten wir in den Datenblättern des Sensors und der RGBs, wie groß die Netzteile sein müssen. Später stellte sich heraus beim Anschließen der RGB-Kette, dass die LED‘s viel mehr Strom in Reihenschaltung benötigten als wir berechnet hatten (über 20A). Folgend beschlossen wir unsere Schaltung um zulöten, wir setzten die Stromversorgung mittig und teilten diese in 4 Reihenschaltungen auf. Wir verringerten den benötigten Stromverbrauch auf ungefähr 10A. Zusätzlich verzichteten wir auf weitere RGBs für das Smiley, um Strom zu sparen und ein transportables Netzteil nutzen zu können.

Eine weitere Herausforderung im Projekt war die Ansteuerung der einzelnen RBGs, da diese teilweise zusammengeschaltet sind und beim Ansteuern einer RBG zwei Stück eingeschaltet werden. Nach mehrfachen neu Löten konnte der Fehler nicht gefunden werden.

Daraufhin haben wir uns eine neue Schaltung (Abb.8) entwickelt, mit der wir unsere Schaltung umsetzen konnten.