STM_Praktikum Sensor D6T-32L-01A

Dieses Projekt bezieht sich auf den Kurs Mikrocontroller an der Universität Bayreuth. Ziel des Kurses ist es, Daten eines Sensors, in diesem Fall eines Infrarotsensors, mithilfe eines Entwicklungsboards auszulesen und auf dem Board oder dem verbundenen Bildschirm wiederzugeben.

Die Ausgabe der Sensorwerte erfolgt etwa alle zwei Sekunden auf dem Display des Entwicklungsboards in Form eines farbigen Bildes. Dabei werden die Temperaturwerte auf einer Farbskala von hoch (rot) zu niedrig (blau) über normal (weiß) abgebildet. Für diese Farbskala sind **zwei Modi** verfügbar, zwischen denen durch **Halten des Joysticks bis zur nächsten Messung** nach rechts für den relativen Modus (REF), oder nach links (Standard) für den absoluten Modus (ABS).

Im **absoluten Modus** bildet die Tiefsttemperatur des Sensors laut Datenblatt (10°C) den unteren Rand der Skala und die Höchsttemperatur (70°C) den oberen. Dazwischen sind die Raumtemperatur (24°) in Weiß und Körpertemperatur (36°) in Orange festgesetzt, um einen stärkeren Ausschlag im alltäglichen Temperaturbereich zu erzielen.

Abbildung 1: die im absoluten Modus verwendete Farbskala

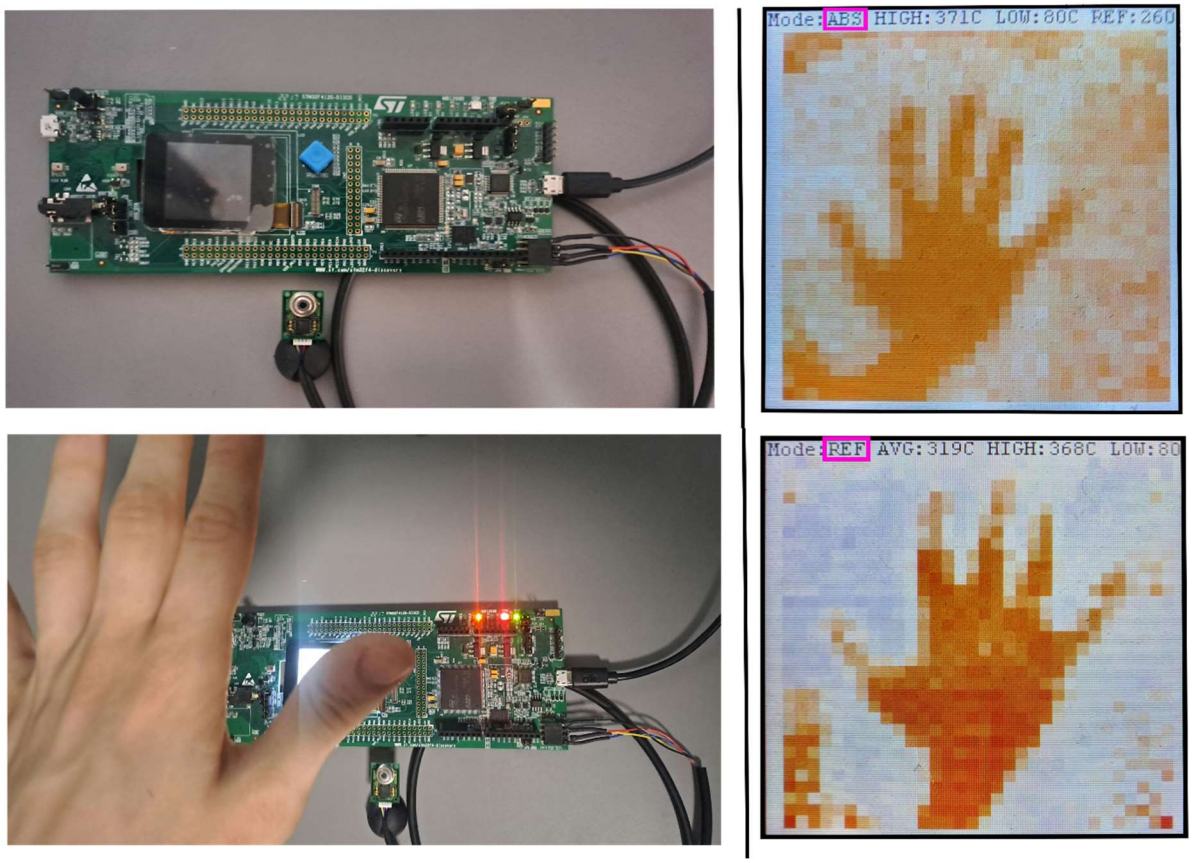
Im **relativen Modus** dagegen wird die tiefste gemessene Temperatur auf Blau gesetzt und die höchste gemessene Temperatur auf Rot. Die Durchschnittstemperatur wird auf Weiß festgelegt und Orange bildet den Durchschnitt + 2,5 °C ab. Dadurch können Objekte unabhängig von der Umgebungstemperatur besser erkannt werden.

In welchem Modus man sich befindet, kann man sowohl an der leuchtenden LED (blau=relativer Modus, orange=absoluter Modus, rot=keine Daten vom Sensor erhalten), als auch anhand des Textes am oberen Bildschirmrand (Abbildung 2: pink umrandet) erkennen. In dieser Zeile stehen je nach Modus auch folgende Werte:

- HIGH: die höchste gemessene Temperatur
- LOW: Die niedrigste gemessene Temperatur
- AVG: Die durchschnittliche Temperatur
- REF: Die Referenztemperatur im Programmcode, die als Weiß abgebildet wird

Diese Temperaturen werden aus Platzgründen jeweils in dem Zehnfachen der Temperatur in Grad Celsius angegeben. Eine angezeigte Temperatur von 371C bedeutet also eine Temperatur von 37,1°C, bzw. $371 \cdot 27{,}415K = 310{,}25$ Kelvin.

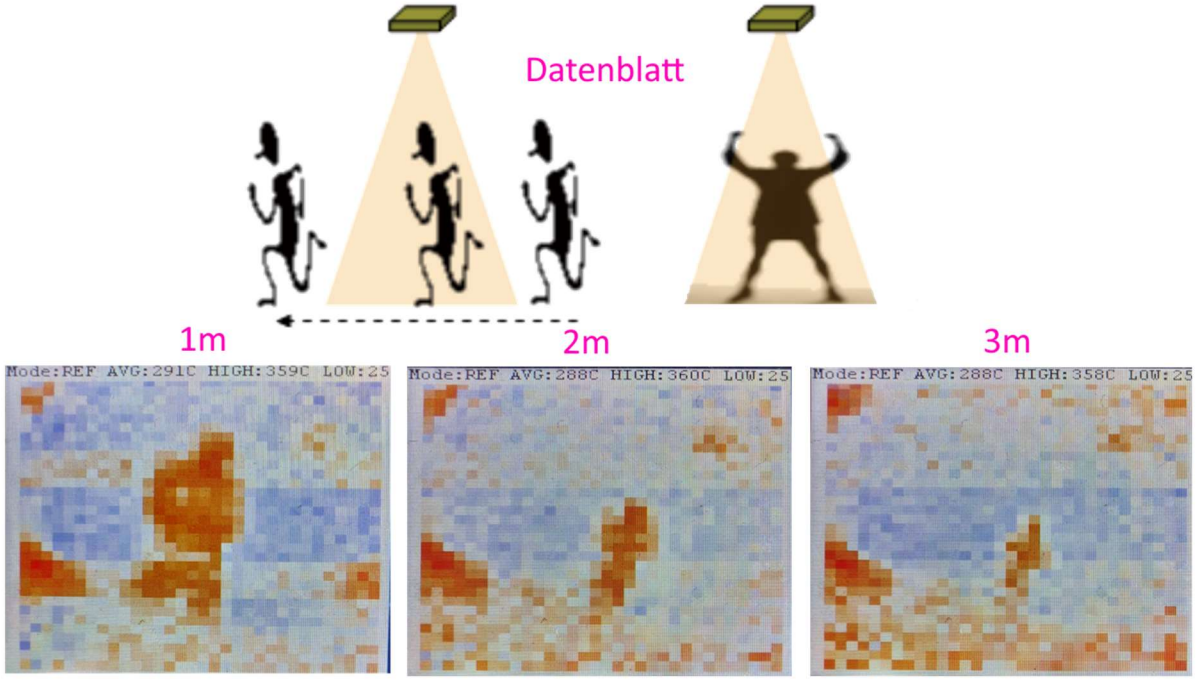
Abbildung 2: Versuchsdurführung mit and über dem Sensor



In der Abbildung ist oben links das Setup mit Board und Sensor, darunter die Durchführung und rechts daneben die Ergebnisse eines Durchlaufs zu sehen, bei dem ich meine Hand über den Sensor gehalten habe. Dabei zeigt das obere rechte Bild das Ergebnis im absoluten, das untere die Temperatur im relativen Modus. Wie man erkennen kann, ist der Kontrast im relativen Modus höher als im absoluten Modus.

Das Datenblatt des D6T schlägt als Anwendung des Sensors das Erkennen von Menschen vor. Für das Erkennen selbst gibt es meinerseits zwar keine Implementierung einer automatischen Erkennung, dennoch halte ich dieses Szenario für ein gutes Beispiel, um die Auflösung des Sensors zu verdeutlichen. Im Folgenden Bild ist zum einen die Abbildung aus dem Datenblatt und darunter meine Messungen, bei denen ich jeweils 1m, 2m, bzw. 3m vom Sensor entfernt stehe. Dabei ist anzumerken, dass bereits auf dem zweiten Bild eine Erkennung durch Software unwahrscheinlich ist, obwohl der D6T-32L-01A der Sensor der D6T Serie mit der höchsten Auflösung ist.

Abbildung 3: Erkennen von Menschen aus verschiedenen Distanzen



Getting started

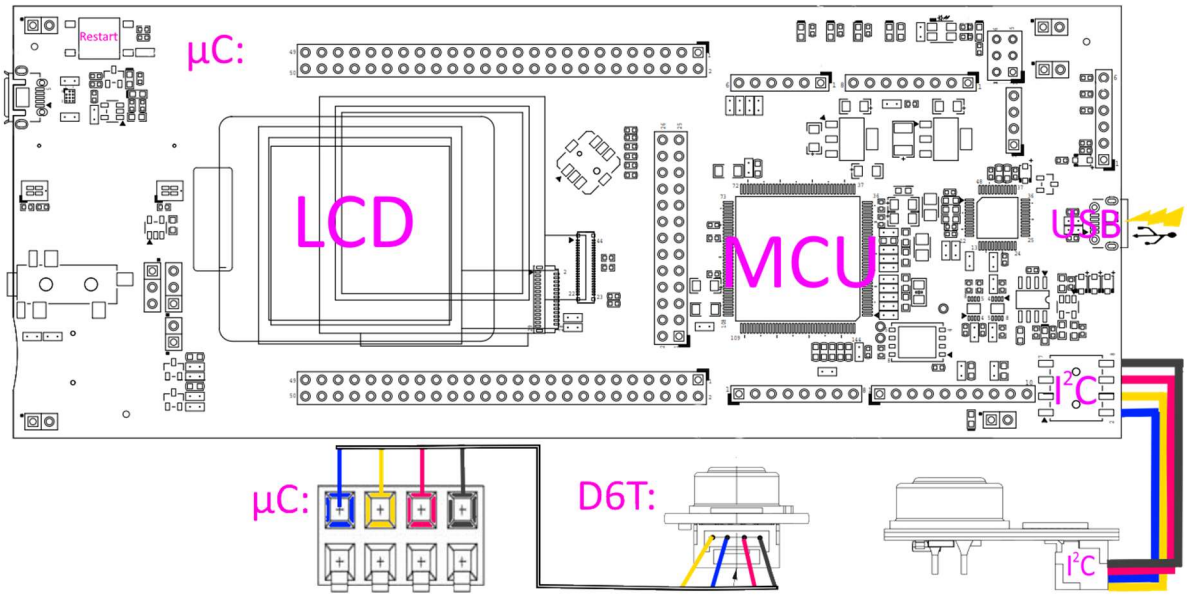
- Für die Durchführung wurde folgendes Setup verwendet:
- Ein D6T-32L-01A IR Sensor, der mit
- einem STM32F412G-Discovery board (hier mit µC abgekürzt) mit
- einem Qwiic zu 4-male Jumper Pin Kabel verbunden ist.
- Ein Mikro-USB-Datenkabel zur Stromversorgung und zum Programmieren des µC wird ebenfalls benötigt.

Für weitere Informationen zu dem Sensor und Mikrocontroller (kurz µC) verweise ich auf die jeweiligen Datenblätter:

- Datenblatt D6T
- Datenblatt STM32

Verkabelung

Abbildung 4: Skizze der Verkabelung des Sensors mit dem Entwicklungsboard



Die Kommunikation zwischen µC und D6T findet über das I^2C Protokoll statt. Der I^2C Port befindet sich auf der dem LCD abgewandten kurzen Seite des µC, unterhalb des USB Connectors, der für die Stromzufuhr und Programmierung des µC verwendet wird. Der I^2C Port des D6T dagegen kann an der Unterseite in Form eines Qwiic Anschluss gefunden werden. Die Farben der in der Abbildung 4 verwendeten Kabel haben die folgende Bedeutung:

* Gelb: SCL (I^2C Clock)

* Blau: SDA (I^2C Datenleitung)

* Rot: VCC (Stromzufuhr)

* Schwarz: GND (Ground)

Software auf dem µC installieren

Für dieses Projekt wurde die frei zum Download verfügbare Software STM32CubeIDE verwendet. Mithilfe dieser Software kann das Projekt auch auf dem Mikrocontroller gespielt werden. Dazu muss das geklonte Archiv als Projekt geöffnet werden, der µC durch ein Datenkabel mit dem Computer verbunden, und der grüne Play-Button am oberen Rand der Oberfläche gedrückt werden.

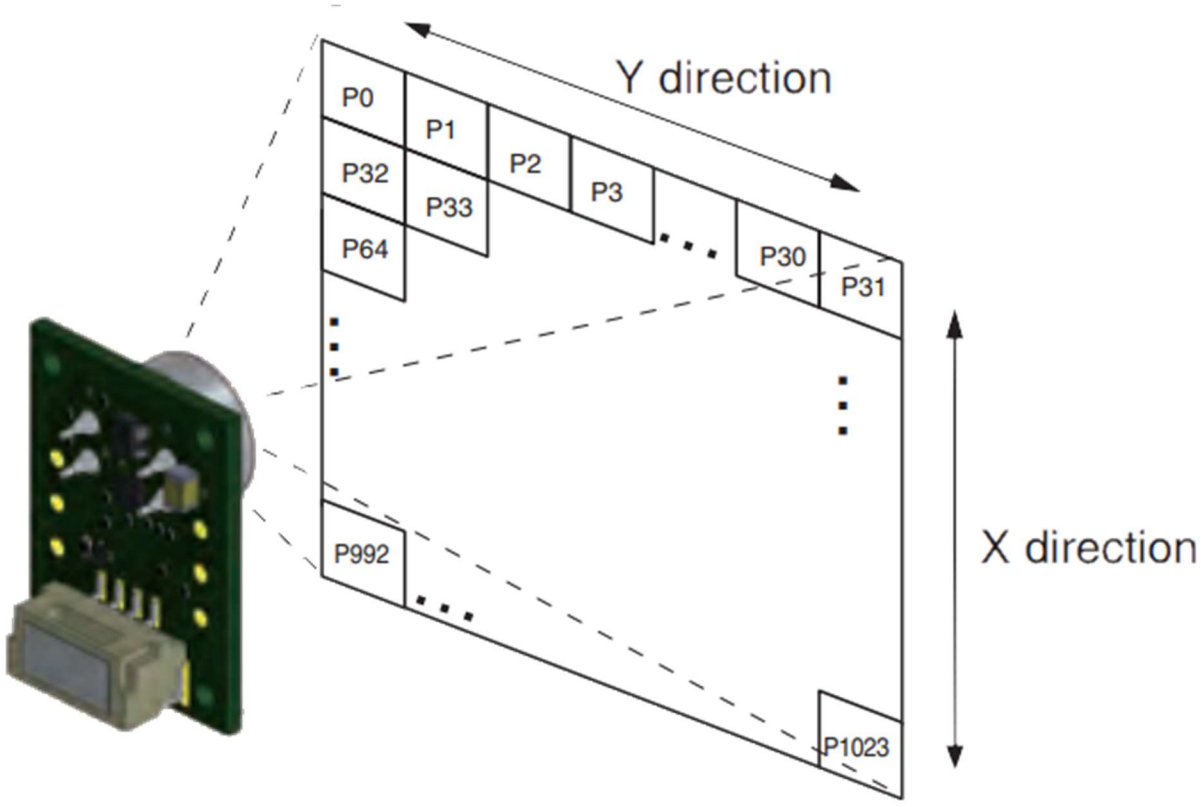
Software zurücksetzen

0. Das Programm beinhaltet bereits einen Watchdog Timer, der etwa alle vier Sekunden das Programm zurücksetzt, sollte dies nicht mehr reagieren. Sollte dennoch das Programm nicht funktionieren, können folgende Schritte versucht werden:
1. Das Neu-Starten des Sensors erfolgt, indem entweder die VCC oder GND Leitung getrennt und neu verbunden werden.
2. Das auf dem Sensor gespielte Programm lässt sich manuell neu starten, indem der Restart-Knopf (siehe Abbildung 4) gedrückt wird.

Funktionsweise der D6T MEMS Sensoren

Für dieses Projekt war der Infrarotsensor D6T-32L-01A gegeben, der 32×32 Temperaturwerte einer rechteckigen Fläche erfasst:

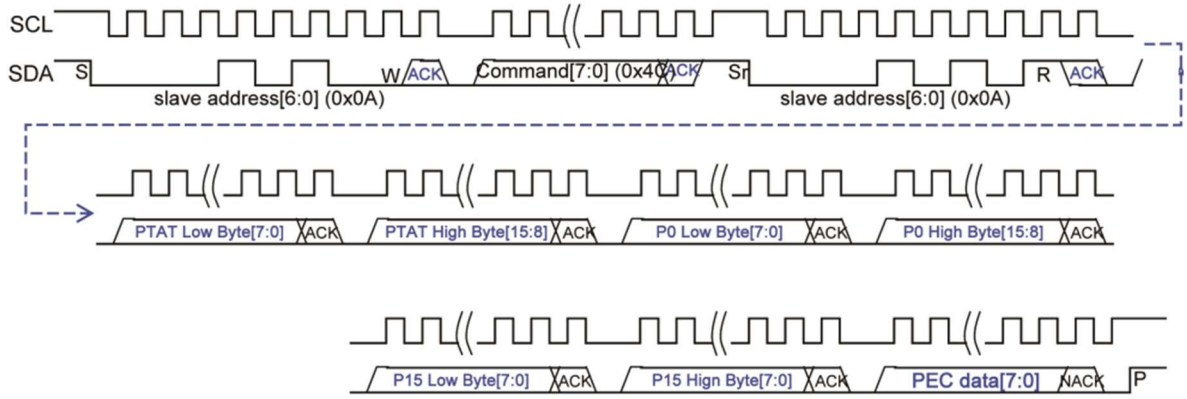
Abbildung 5: Abtastbereich des Sensors



Auf diese Temperaturwerte kann über das I^2C Protokoll zugegriffen werden. Das I^2C überträgt Daten in digitaler Form, über zwei Leitungen. Dabei wird eine als Pulsgeber (Clock) und die andere als Datenleitung genutzt. Die Kommunikation mittels I^2C erfolgt

bidirektional nach dem Master-Slave-Prinzip, wobei der Mikrocontroller den Master und der Sensor den Slave darstellt und hat immer folgenden Aufbau:

Abbildung 6: Kommunikation mit dem Sensor über I^2C



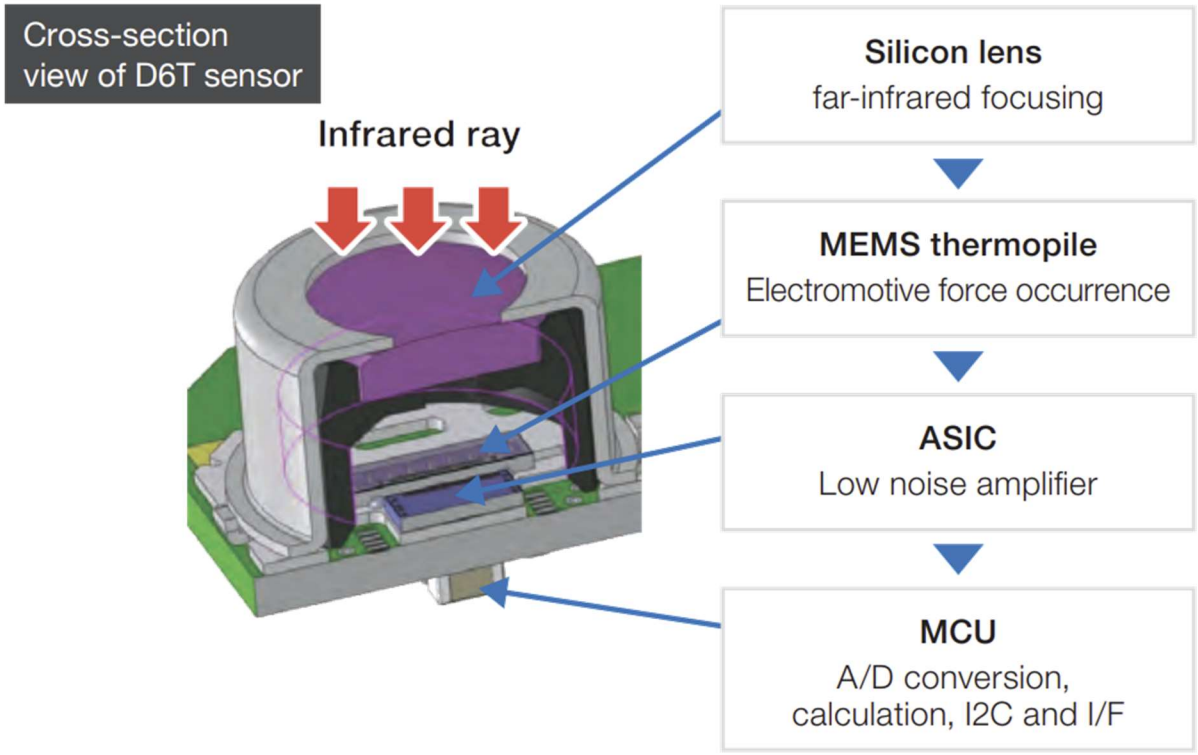
1. Startbedingung S: Die Kommunikation beginnt mit einer Startbedingung (eine Änderung des SDA (Serial Data Line) von HIGH auf LOW).
2. Adressierung: Der Master sendet die Adresse des Slave, für den D6T ist das `0x14`, zusammen mit einem Bit, das angibt, ob es sich eine Lese- oder Schreiboperation handelt.
3. Datenübertragung: Der Slave sendet eine Bestätigung und die Datenübertragung beginnt.
4. Stopbedingung: Die Kommunikation endet mit einer Stopbedingung, die vom Master gesendet wird (Änderung des SDA von LOW auf HIGH).

Jede Kommunikation auf der Datenleitung erhält in jedem Schritt von der anderen Seite ein Acknolage (ACK)

Da das I^2C Protokoll lediglich 8-Bit-Daten unterstützt, müssen die 16-Bit-Temperaturwerte in HIGH- und LOW-Bits übertragen werden. Diese werden dann auf dem µC wieder zu einem 16Bit Temperaturwert zusammengesetzt. Diese Werte beschreiben in diesem Fall die 10-fache Temperatur.

Dabei werden die Temperaturwerte im Sensor durch das folgende Setup gemessen:

Abbildung 7: Aufbau des Temperatursensor



Die einfallende Infrarot-Strahlung wird durch eine Silikon-Linse auf einen Thermophilsensor gebündelt und es wird die dabei resultierende Kraft gemessen. Durch den Vergleich mit einer internen Lookup-Tabelle wird die Temperatur des Infrarot-Strahlen emittierenden Objekts ermittelt. Diese Werte können dann über das I^2C Protokoll ausgelesen werden.

Fehlerabschätzung

Die Fehlerabschätzung habe ich anhand einer Messung der Handwärme mit Haushaltsmitteln durchgeführt. Dafür habe ich sowohl den Sensor als auch ein Domotherm Thermometer vollkommen in meiner linken Hand umschlossen und die Temperaturwerte verglichen. Das Thermometer hat Werte von $36,78\pm 0,18$°C (6 Messungen) gemessen, wohingegen der Sensor $37, 78\pm 0,62$°C (6 Messungen über 6 Sekunden) gemessen hat. Damit ist eine Abweichung von etwa 1 bis 1,5°C zu erwarten.

Softwarearchitektur und Designentscheidungen

In den vorherigen Kapiteln wurde auf die Zusammensetzung und Verwendung des Sensors in Verbindung mit dem Programm eingegangen. Dieses Kapitel dagegen soll einen Überblick über die Struktur des Programmcodes geben. Das Programm selbst, von dem hier die Rede ist, lässt sich in dem Ordner „/Core/Src/sensor“ wiederfinden, wobei als Einstiegspunkt die Methode `d6t.h/d6t_reading_to_lcd` aufgerufen wird. Diese Methode dient zur Koordinierung der Kommunikation und der Anzeige auf dem LCD. An dieser Stelle wird auch auf einen Wechsel der Modi überprüft und die entsprechende Render Methode gewählt.

Aber bevor Sensorwerte gerendert werden können, muss eine Kommunikation mit dem Sensor hergestellt werden. In der Datei „communication.h“ werden die entsprechenden Methoden und Konstanten (wie die Sensoradresse) spezifiziert, die für eine Kommunikation mit dem Sensor und Umwandlung der Rohdaten in °C notwendig sind. Wie in der Funktionsweise erwähnt, findet die Kommunikation über das I^2C Protokoll statt. Dabei muss dem Empfangen der Daten der Befehl 0x4C vorweggehen, damit die Daten in das Übertragungsregister geschoben werden. Die Daten selbst aktualisiert das integrierte Board alle 300ms. Die Abfrage der Daten geschieht aber nur etwa ein mal pro Sekunde, da mir aufgefallen ist, dass sich bei einem kürzeren Delay sehr oft deutliche Streifen bilden, die nicht verschwinden. Bei starken Änderungen sind auch im derzeitigen Zustand leichte Streifen erkennbar, diese verschwinden aber bei den nächsten Render-Durchläufen.

Ich habe mich für eine Umrechnung der Rohdaten in Celsius entschieden, da der Sensor selbst bereits Celsius liefert und die Anzeige hauptsächlich über Farben geschieht. Bei der textuellen Anzeige habe ich mich auch gegen eine Umrechnung in die SI-Einheit Kelvin, bzw. eine Gleitkommazahl entschieden, weil mein Platz auf dem Bildschirm sehr limitiert ist. Aus diesem Grund zeige ich die Temperaturwerte auch als Farben anstelle von Zahlen an, da 32x32 mindestens dreistellige (Celsius), wenn nicht sogar vierstellige (Kelvin) Werte auf einem 240×240 Display gerade einmal 7 Pixel pro Wert Platz hätten.

Die Funktionen zum Rendern der Werte befinden sich ausgelagert in der „render.h“. Das Rendern beinhaltet dabei das Interpolieren der Temperatur zwischen Grenzen, das Übertragen der Farbwerte auf ein Farbspektrum und schließlich das Anzeigen der Farbwerte auf dem Bildschirm. Für die Interpolation habe ich zwei verschiedene Modi erstellt, da mir an warmen Tagen aufgefallen ist, dass man auf einem weitgehend orangen Bildschirm wenig erkennen kann. Dennoch ist es für Messungen oft wichtig, einen festen Vergleichswert beizubehalten, deswegen gibt es die Aufspaltung in diesen zwei Modi. Für den absoluten Modus habe ich die Temperaturwerte für Weiß und

Orange auf Zimmertemperatur und Körpertemperatur gewählt, da die meisten Temperaturen < 70° im Alltag in diesem Bereich liegen.

Für die Farbskala habe ich einen Übergang von Blau zu Rot aufgrund der allgemein verbreiteten Synästhesie, dass Blau/Weiß als kalt und Rot/Orange als warm wahrgenommen wird, gewählt.

Easter Egg: Snake

Wenn man einen Blick in den Programmcode wirft, fällt auf, dass neben dem Ordner `/sensor` ein weiterer Ordner mit der Aufschrift `/snake` im Projektverzeichnis liegt. Dieser Ordner enthält den Programmcode, um das Spiel Snake auf dem Mikrocontroller zu spielen. Dieses Spiel habe ich in der Zeit zwischen der Ausgabe der Mikrocontroller und der Ausgabe der Sensoren als ein erstes Hello World!-Programm entwickelt. Auch wenn ich das Spiel eigentlich nur zu Lernzwecken erstellt habe, entschied ich mich dennoch, es meiner Abgabe beizufügen, da ich es in der genannten Zeit fertiggestellt hatte und die Einbindung mit wenig Aufwand verbunden war. Das Starten des Spiels auf dem Entwicklungsboard ist nur zum Programmstart/Programm-reset durch das Halten des blauen Joysticks nach oben möglich. Sobald man sein Spiel beendet hat, startet das Programm zum Auslesen des Sensors.

Sensor Timeout state

Sensoren der D6T-Serie haben einen sogenannten Timeout state. Dieser wird aktiviert, wenn über einen Zeitraum von 75ms bei laufender Stromzufuhr die Clock aussetzt. Sobald ein Sensor in einem Timeout State ist, kann dieser Zustand erst wieder durch ein Trennen und Neuverbinden der GND- oder VCC-Leitungen (Schwarz/Rot) verlassen werden.

Dieser Fehler kann im Programm auftreten, wenn das Programm zu einem ungünstigen Zeitpunkt bzw. in schneller Abfolge durch den auf dem Board verbauten Reset zurückgesetzt wird. In diesem Fall blinkt die rote LED und ein Text wird auf dem Bildschirm angezeigt. Zum Beheben des Problems muss die Verbindung zum Sensor unterbrochen und neu hergestellt werden. Danach sollte das Programm durch den Watchdog neu starten und wie beschrieben funktionieren.

Quellen

HAL Dokumentation

Datenblatt Entwicklungsboard

Warme-/kalte Farben

Datenblätter Sensor:

- Datenblatt D6T EN
- Datenblatt D6T

Autor

Yannick Pahlke 1841500 (@Reskuzo auf Github)