Technische Dokumentation

Morse Messenger

.. - ... / -..-. / -- --- ... -.--

Sommersemester 2019
für die Kurse IT-Systeme und Mobile Systeme
bei Prof. Dr. Torsten Edeler und Prof. Dr. Andreas Plaß
von Jan Jessen (2202032), Maryam Safi (2224027), Malte Saupe (2315546)
und Caroline Wolf (2320493)

Inhalt

Einleitung

Wir haben eine Messenger-App mit Morse-Protokoll realisiert. Eine Android-App sendet den eingegebenen Text per Bluetooth an einen Arduino, welcher ihn in Morse-Code moduliert und mittels einer handelsüblichen Taschenlampe sendet. Ein identisches Set aus Arduino und Smartphone empfängt das Signal mittels eines Photoresistors, dekodiert es und sendet es an ein zweites Handy. Somit ist asymmetrische Zwei-Wege-Übertragung möglich.

Programmierung des Arduinos

Es gibt viele Wege den computerisierten Empfang von Morse zu vereinfachen indem man vom Standard abweicht. Viele Implementationen online nutzen zwei diskrete Kabel, einen festgeschriebenen Takt oder padden jeden Buchstabencode auf 8 Zeichen.
Fokus dieses Projektes war daher, genau das nicht zu tun. Der Takt ist vom Sender einseitig frei wählbar und es werden echte Morse-Kontrollcharaktere genutzt. Es wird ein analoges Lichtsignal gelesen und interpretiert.
Da der Arduino sowohl Senden als auch Empfangen kann wird in jedem Loop-Durchlauf überprüft, ob gerade ein Flankenwechsel am Sensor oder eine zu sendende Nachricht vorliegt. Ist ersteres der Fall wird mit dem Senden des eigenen Strings bis zum Empfangsende gewartet.

Senden:

Die Stringannahme und Vorverarbeitung erfolgt in der Funktion updateBT(). Erwartet wird eine bereits auf verbotene Symbole gefilterte Nachricht. Direkt nach dem Input wird auf mögliche Kontrollsignale geprüft, diese beinhalten zur Identifizierung einen Doppelpunkt, ein sonst illegales Zeichen. Ein Kontrollsignal kann z.B. die Änderung der Sendefrequenz mit “CLK:XX” sein.
Ist der Input kein Kontrollsignal, wird er um Start und Endzeichen erweitert und mit den Funktionen wordSend() und charSend() übertragen. Nach vollständigem Durchlauf gibt der Arduino eine Sendebestätigung an das sendende Handy („SENT:”).

Empfangen:

Unser Dekodieralgorithmus interpretiert den Sensorinput nicht selbst sondern arbeitet komplett digital. Daher muss im ersten Schritt das analoge, rauschige Signal des Photoresistors korrekt interpretiert werden. Dies geschieht in der Funktion updateSensor(): Weder das absolute Signallevel noch die Signalamplitude kann als bekannt vorausgesetzt werden. Ersteres hängt vom Umgebungslichtlevel ab, letztere von der Sendedistanz. Die Lösung sind zwei ständig gegen das Signal dekrementierte Zählvariablen, minRx und maxRx, die den minimalen und maximalen Signalwert speichern. Sie bilden so eine Hüllkurve deren Mittelwert (divider) die Detektionsschwelle für einen Flankenwechsel ist.

Wurde ein Flankenwechsel erkannt, wird die Funktion interpret() aufgerufen. Sie berechnet den vermuteten Sendertakt laufend neu und interpretiert das Signal. Erkannte Punkte und Striche werden zum späteren Übersetzen im String receivedLetter gespeichert und am Buchstabenende mit translateLetter() übersetzt. Nach Empfang eines validen Encharakters wird ein Acknowledge-Signal als Empfangsbestätigung gesendet („ACK:“) .

Standby:

Die Funktion checkStatus() wird in jedem Loop-Durchlauf aufgerufen und prüft das Bestehen der Verbindung zwischen den Arduinos. Alle ca. 45 Sekunden (leicht randomisiert) wird eine bitte um Acknowledge gesendet und in die ackActive Variable geschrieben. Falls kein valides Acknowledge vom Empfänger kommt, fragt der Sender in 10s-Abständen erneut. Der Status wird an zwei Debug-LEDs angezeigt, so ist ein eventueller Fehler schnell auf Hardware (kein receive) oder Software (kein acknowledge) einzugrenzen.

Hardware

Für die Realisierung wurden folgende Geräte benötigt:

- 2x Arduino - 4duino Wireless Modul HC-06
- 2x Arduino Mega 2560, ATmega 2560, USB
- 2x Seeed Studio Grove - Mega Shield v1.2 Arduino Mega Grove compatible
- 2x Android-Smartphones
- 2x Relay helishun hls8l-dc12v-s-c
- 2x Readaeer CREE T6 Taschenlampen
- 2x POPESQ® Photoresistor 5mm
- Div. Widerstände und Kondensatoren

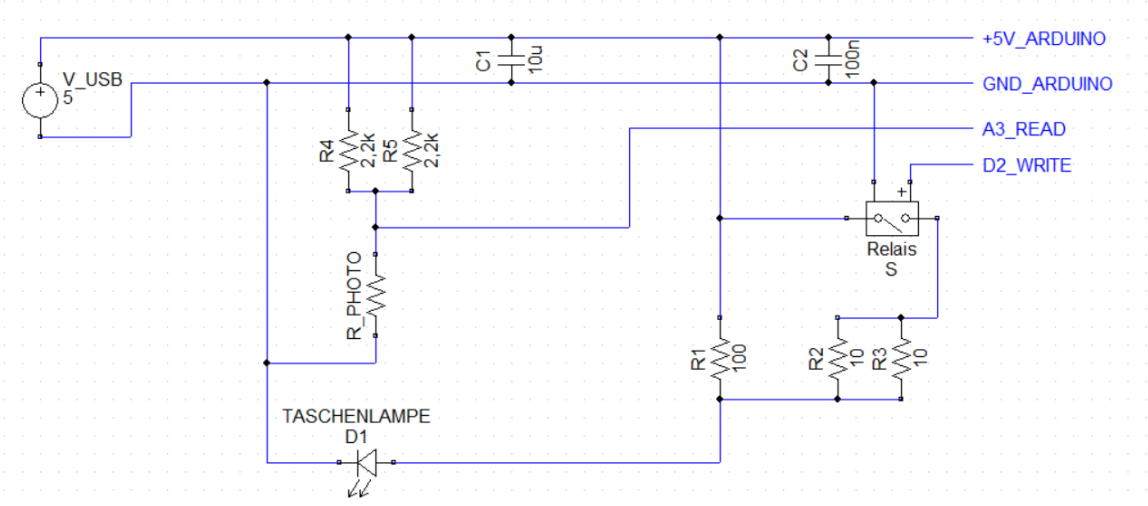
Bauteile:

Zunächst haben wir statt Taschenlampen die Schaltung mit LED Panels (Adafruit Neopixel, Neomatrix 8x8) aufgebaut. Diese konnten allerdings die Signale nicht schnell genug verarbeiten, wodurch Farbfehler entstanden. Außerdem war das erzeugte Licht nicht gerichtet. Um eine bessere Reichweite zu erreichen, haben wir nach einiger Suche zwei günstige, zoombare Taschenlampen angeschafft.

Die ursprünglich angeschafften sensiblen Lichtsensoren konnten aus Geschwindigkeitsgründen leider nicht verwendet werden, sie lieferten einen Wert maximal alle 40ms und waren durch die I2C-Auslesung sehr inkonsistent im Timing.

Schaltung:

Die finale Schaltung nutzt also zur Signaldetektion einen Photoresistor einfachster Bauart, ausgelesen per Spannungsteiler und direktem Anschluss an einen Arduino-Analogpin. Die Steuerung der Taschenlampe erfolgt durch eine relaisgesteuerte Parallelschaltung eines kleinen Widerstands zu einem 100 Ohm Widerstand. Ein komplettes „aus" triggert diverse Funktionen der Taschenlampe, durch konstante schwache Versorgung wird das übergangen. Gleichzeitig wird so das Ausrichten des Strahls vereinfacht. Die Kondensatoren C1 und C2 kompensieren ein beobachtetes Ground-Bounce-Phänomen.



App Programmierung

Die App wurde in Kotlin mit AndroidStudio realisiert. Beim Starten der App öffnet sich die MainActivity. Hier kann sich der Nutzer via Bluetooth mit dem Arduino verbinden und einen Nutzernamen eingegeben. Wenn der Nutzer auf den „Zum Chat"-Button drückt, kommt dieser in die ChatActivity, wo er Nachrichten senden und empfangen kann. Es gibt hier die Möglichkeit, die Übertragungsgeschwindigkeit der Morsezeichen zu ändern.

OptionsMenuActivity

Hier wird im Menu ein Optionen-Icon hinzugefügt und für den Klick auf das Icon die Methode showRulesDialog() implementiert. Diese Methode initialisiert den Dialog, der die Regeln für das Morsen im Morse Messenger enthält. Die Activity vererbt ihre Methoden an MainActivity und ChatActivity, sodass von dort aus das Menü geöffnet und der Dialog mit den Regeln dargestellt werden kann.

BluetoothConnectionService

Dieser Service sorgt dafür, dass die Bluetooth-Verbindung von der MainActivity in die ChatActivity weitergegeben wird. Es wird in beiden Activities ein Objekt dieser Klasse erzeugt und die Methoden setBt() und getBt() für die Übergabe der Verbindung genutzt.

BluetoothService, ConnectThread und ConnectedThread

Die Klassen aus der Vorlesung, um die BluetoothDevices in der Umgebung zu finden, eine Verbindung herzustellen (ConnectThread) und Nachrichten zu senden/empfangen (ConnectedThread). BluetoothService wurde um die Funktion checkSpecialCharacters(input: String) erweitert. Diese Methode filtert die Sonderzeichen, die im Morse-Alphabet nicht verfügbar sind, aus den zu sendenden Nachrichten.

MainActivity

Ein Objekt von BluetoothService wird erstellt und auf dessen Methoden zugegriffen, um eine Bluetooth-Verbindung mit dem Arduino herzustellen. Beim Finden eines Geräts in der Umgebung, wird der DevicesAdapter aufgerufen, um die RecyclerView zu füllen. Beim Starten der ChatActivity wird per Intent der Username an diese weitergegeben.

DevicesAdapter und ViewHolder

Der Adapter verbindet die BluetoothDevice-Daten aus der MainActivity mit der RecyclerView. onCreateViewHolder initialisiert den ViewHolder. onBindViewHolder wird für jeden ViewHolder (jedes BluetoothDevice) aufgerufen. Hier wird die Methode bind() des ViewHolder aufgerufen. Im ViewHolder ist bind() definiert. Eine View (bluetooth_list_item.xml) wird mit den Daten aus dem BluetoothDevice gefüllt und das BluetoothDevice mit einem onClickListener verknüpft.

ChatActivity

Ein Objekt von BluetoothService wird erstellt. In einem EditText-Feld können Nachrichten verfasst und per Send-Button und bluetoothService.write() an den Arduino geschickt werden. Ein Objekt Message wird erstellt und an den MessageAdapter weitergeleitet, um die Nachricht im Chat-Fenster anzuzeigen. In receiveMessageFromOtherDevice() wird die Art der Nachricht festgestellt (SENT: fertig gesendet, ACK: beim Empfänger angekommen, USR: Nutzername).

MessageAdapter und ViewHolder

Der Adapter verbindet die Message-Daten aus der ChatActivity mit der RecyclerView. In onBindViewHolder wird anhand der Message-id festgestellt, ob die Nachricht vom Sender oder Empfänger kommt und anhand dessen eine View (message_list_item.xml) mit den Daten aus dem

Message-Objekt gefüllt. Im Adapter wird außerdem in showSENTArrow() und showRECEIVEDArrow() der Sende-Status der Nachricht aktualisiert.

Message

Diese Klasse wird verwendet, um in der ChatActivity Objekte der Nachrichten von Sender und Empfänger zu erzeugen. Die id bestimmt, ob die Nachricht von Sender (0) oder Empfänger (1) ist.

Im folgenden UML-Klassendiagramm sieht man den groben Aufbau: