Hochschule RheinMain

Fachbereich Ingenieurwissenschaften
Studiengang Angewandte Physik
Studienrichtung Physikalische Technik

Laborbericht

Resistives Touch-Panel

LV: Embeeded System Labor

Versuchsdurchführung: 4. Februrar 2022

Studierende Dennis Hunter ()
Tim-Jonas Wechler (1137877)

Rüsselsheim am Main, 16. Februar 2022

Inhaltsverzeichnis

1 Einleitung

Seit einigen Jahren findet man immer häufiger Bedienungen mit Touchscreens. Ihr Einsatzbereich scheint keine Grenzen zu haben und so hat sich die Technologie in den letzten Jahren rasant weiterentwickelt.

1.1 Ziel des Projekts

In dieser Projektarbeit ist es Ziel, die Aufnahme und Auswertung der Koordinaten mittels eines resistiven Touchscreen. Zum Schluss soll die Steuerung noch dahingehend erweitert werden, dass damit die Maus eines PC's gesteuert werden kann.

1.2 Grundlagen

1.2.1 Resistives Touchscreen

Es gibt bei den resistiven Touchscreens zwei unterschiedliche Gruppen. Zum einen gibt es 4-Wire resistive Touchscreens. Diese haben vier Anschlüsse, über die die Positionsauswertung läuft. Zudem gibt es noch die 5-Wire resistive Touchscreens. Hier sind fünf Anschlüsse, über die die Positionsauswertung durchgeführt wird.

Bei einem 4-Wire resistiven Touchscreen gibt es zwei Ebenen, bei denen die obere auf die untere gedrückt werden kann. Eine Ebene ist mit Elektronen geladen und über die andere Ebene misst man die Spannung die in eine Richtung bei der Berührung abfällt. Für die andere Richtung ist es gespiegelt. Die Ebene, die die Spannung in eine Richtung misst, ist für die Messung in die andere Richtung mit Elektronen geladen. Die Messung wird dann von der anderen Ebene durchgeführt. Über die unterschiedlichen Beträge der Spannung lässt sich dann die Koordinate in x-Richtung und y-Richtung bestimmen.

Die 5-Wire resistive Touchscreens haben ebenfalls zwei Ebene. Der Unterschied zu den 4-Wire Touchscreen liegt darin, dass es nur eine Ebene gibt, die geladen ist. Normalerweise ist es die untere Schicht. Die obere Schicht misst für x-Richtung und y-Richtung die abfallende Spannung. Auch hier wird durch Druck auf die obere Schicht ein Kontakt zwischen den beiden Ebenen erstellt, damit die Messung durchgeführt werden kann.

Die 5-Wire Touchscreens sind gegenüber den 4-Wire Widerstandsfähiger und haben dadurch eine längere Lebenserwartung.[Mil]

In diesem Projekt wird ein 4-Wire resistiver Touchscreen von der Firma Fujitsu verwendet.

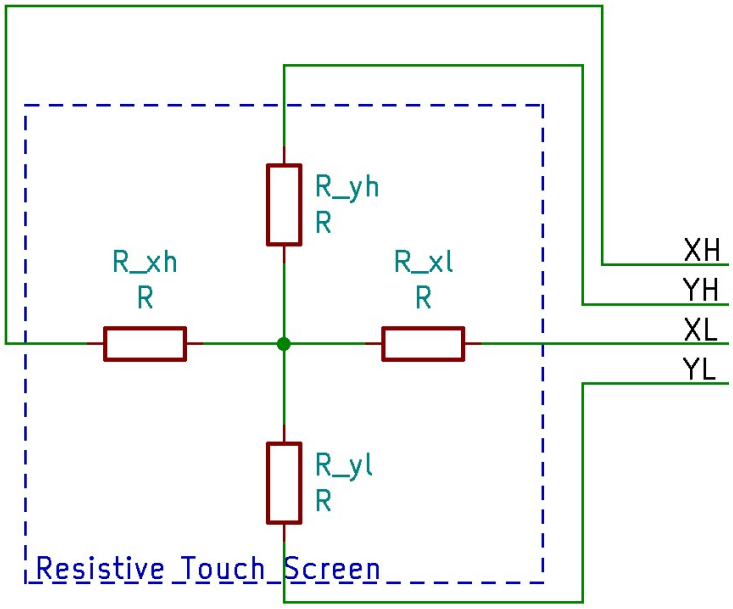


Abbildung 1.1: Schemadarstellung eines 4-Wire resistiven Touchscreen

Der verwendete Touchscreen besitzt für jede Richtung (x und y) jeweils zwei Anschlüsse fig. 1.1. In x, wie auch in y-Richtung sind jeweils 2 Widerstände eingezeichnet. Diese sollen den Widerstand auf der jeweiligen Ebene, über die die Spannung abfällt, darstellen. Die Aufnahme der Werte für einen Koordinatenwert wird über das Prinzip des Spannungsteiler realisiert.

1.2.2 Arduino Leonardo Board

Die Steuerung des Touchscreens wird über das Arduino Leonardo Board realisiert. Dieses hat im Vergleich zum Arduino Uno Board die Möglichkeit sich als Peripherie an einem PC an zu melden. Ermöglicht wird dies durch den Mikrocontroller ATmega32u4. Der Mikrocontroller arbeitet nicht mit einem USB-Chip, sonder verarbeitet intern die serielle eingehende Daten und konvertiert diese für die Nutzung der USB-Schnittstelle.

2 Versuchsaufbau

Das Porjekt wird fertig zusammengebaut ausgehändigt.

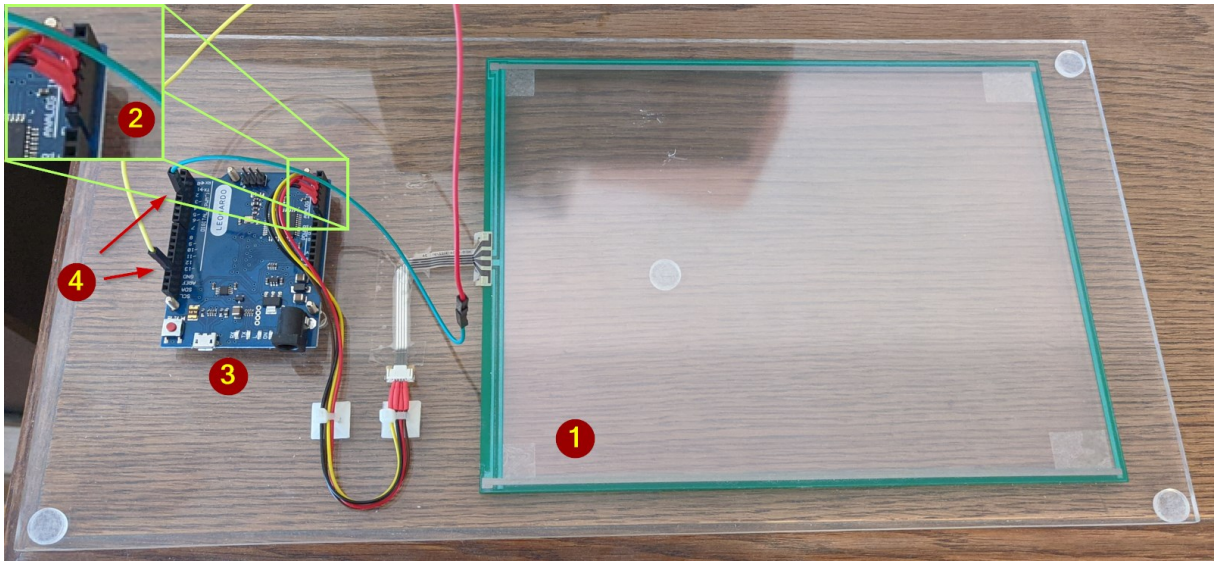


Abbildung 2.1: Versuchsaufbau

1 Touchscreen

2 Anschlüsse des Touchscreen am Leonardo Board

3 Arduino Leonardo

4 Pins für die Steuerung, ob man die Maus bedienen möchte oder nicht

3 Implementierung

Aus den oben beschrieben Überlegungen und dem daraus entstandenen, in fig. 4.3 gezeigten grundlegenden Programmablauf wird folgt die nachfolgend beschriebene Implementierung in Software.

3.1 Vorbereitung der MCU

Zur Umsetzung werden die Timer `Timer0` und

4 Aufnahme von Koordinatenpunkte

4.1 Problem

Wenn ein Koordinatenpunkt aufgenommen werden will, gibt es mehrere Probleme auf die man stößt.

Zum einen kann nicht gleichzeitig die X und Y Komponente der Koordinate bestimmt werden. Dies muss nacheinander folgen. Grund hierfür ist dem Aufbau des Touchscreens geschuldet.

Bei der Inbetriebnahme wird sich noch ein weiteres Problem ergeben. Falls der Touchscreen nicht betätigt wird, gibt der Mikrocontroller trotzdem Werte aus. Dabei handelt es sich um Werte die keinen Sinn ergeben und die Steuerung der Maus z.B. bei einer Ruhephase stören würden.

Bei einer Messreihe kann es vereinzelt zu einem Wertesprung kommen. Dieses Problem ist der Messunsicherheit des Systems geschuldet. Diese Sprünge gilt es aus der Messreihe heraus zu filter und zu glätten.

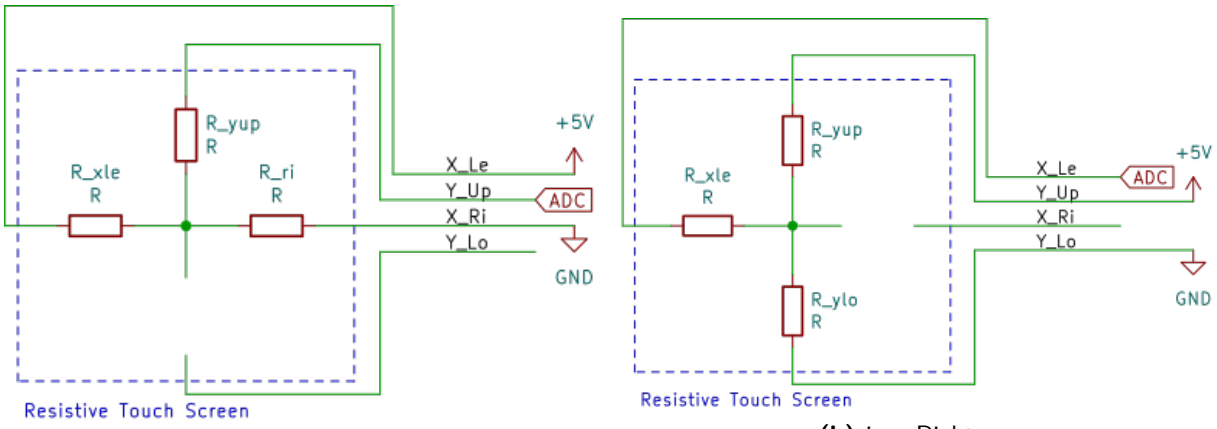
4.2 Lösungsansatz

Um das Problem, bei einer nicht Betätigung des Touchscreens, zu lösen, soll vor einer Aufnahme von Koordinatenwerte geprüft werden, ob der Touchscreen betätigt wird. Ist dies der Fall so soll die Messung durchgeführt werden.

Um eine Koordinatenkomponente zu bestimmen, benötigt man drei Anschlüsse des Touchscreens. Zwei davon sind in der Richtung die man messen möchte und der dritte Anschluss ist einer der beiden übrigen Anschlüsse. Mit diesem wird der Spannungsteiler auf gespannt um den Wert der Koordinate zu bestimmen.

In den fig. 4.1 auf fig. 4.1 wird dieser Lösungsansatz veranschaulicht. Bei dem Lesen der x-Komponente fig. 4.1a wird der Pin `X_Le` (steht für X-Links) auf eine Spannung von 5 V gesetzt. Der Pin `X_Ri` (steht für X-Rechts) wird auf 0 V gezogen. Der Pin `Y_Up` (steht für Y-Oben) wird auf den Modus `Hi Z` (Hohe Eingangsimpedanz) gesetzt. In diesem Modus fällt der Widerstand in die y-Richtung so klein aus, das er vernachlässigbar gering ist. Um die y-Komponente zu lesen wird das wie in fig. 4.1b auf page 6 der Pin `Y_Up` auf 5 V gesetzt und der Pin `Y_Lo` (steht für Y-Unten) auf 0 V gesetzt. Hier wird der `Hi Z` Modus auf den Pin `X_Le` gesetzt.

Um nun noch eine Messunsicherheiten aus zu filtern sollen, bei der Messung einer Koordinatenkomponente, mehrere Messpunkte aufgenommen werden. Diese werden anschließend über ein Filter-Funktion ausgewertet. Der Wert der bei der Auswertung als Ergebnis herauskommt, wird als gemessene Koordinatenkomponente ausgegeben. Das Schaltbild hierzu ist in fig. 4.2 zu sehen.



(a) in x-Richtung **(b)** in y-Richtung

Abbildung 4.1: Schaltbild für das Messen der Koordinatenpunkte

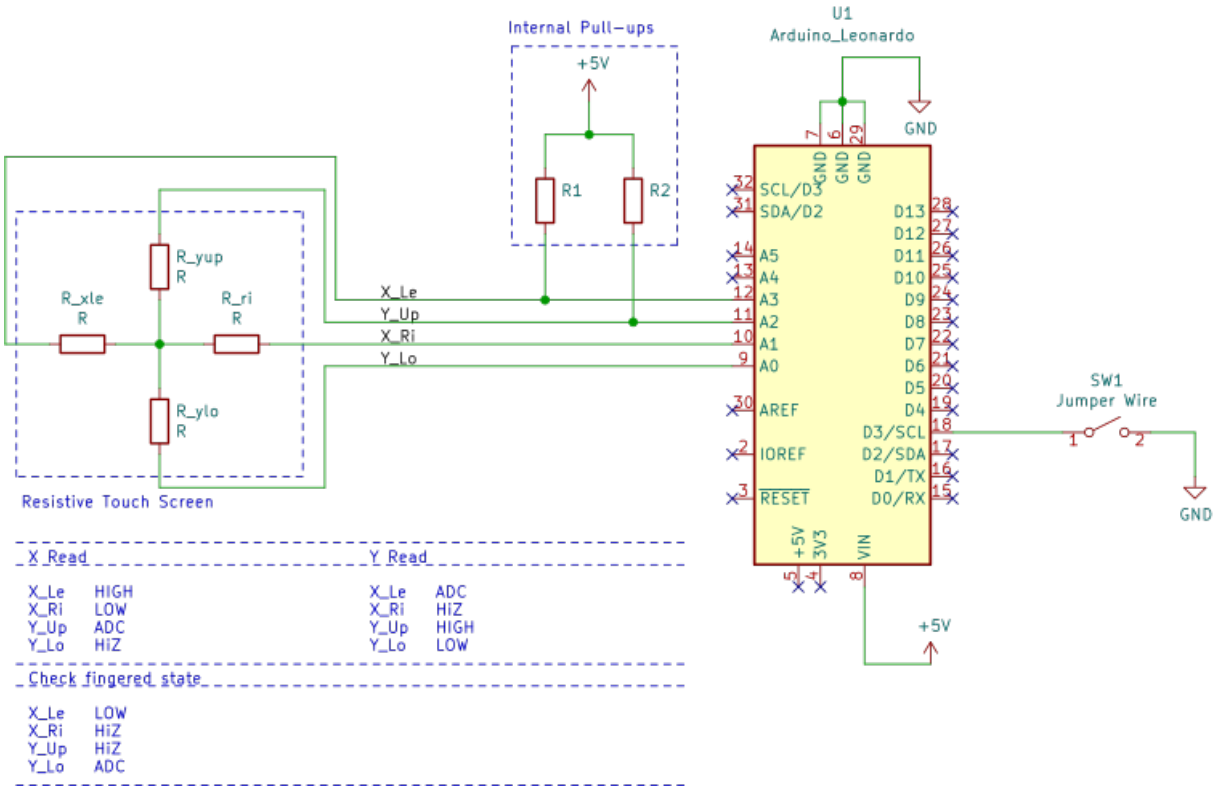


Abbildung 4.2: Schaltbild des Projekts

Die einzelne Lösungsansätze werden in der Arduino-Umgebung umgesetzt. Der Programmablauf ist in fig. 4.3 als Flow-Chart dargestellt.

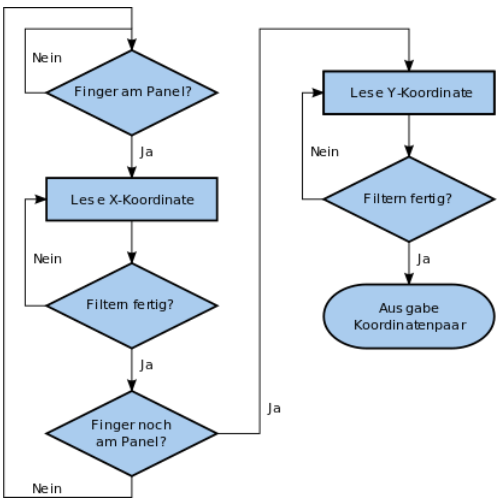


Abbildung 4.3: Darstellung des Programmablaufs

5 Untersuchung der Messdaten

Um eine Aussage über die Qualität des Touchscreen treffen zu können, werden mehrere Untersuchungen angestellt.

Bei der ersten Untersuchung wird auf die Mitte des Touchscreen gedrückt und die Position wird für eine gewisse Zeit gehalten. Die Daten werden anschließen ausgewertet (siehe section 5.1).

Um die erste Untersuchung zu erweitern wird nun die Mitte des Touchscreen wiederholt gedrückt. Hier bei soll die Wiederholbarkeit eines Punktes auf dem Touchscreen untersucht werden. Die Auswertung ist in section 5.2 zu finden.

Bei der letzten Untersuchung wird die Linearität des Touchscreen untersucht. Hierfür gibt der Hersteller eine Garantie, unter der sich die Linearität des Touchscreen befinden soll. Den Wert der angegeben wird liegt bei 1,5 % (siehe appendix A.1 Seite 3 des Datenblatts).

Um die Nachfolgende Untersuchungen korrekt durchführen zu können muss zu nächst der Reaktionsbereich des Touchscreens ermittelt werden. Durch seine Bauform hat dies einen Randbereich an dem es nicht zuverlässig Werte ausgibt. Umkehrschluss, das Programm erkennt nicht das etwas den Touchscreen betätigt. Im Datenblatt werden Werte für den Bereich genannt in dem es zuverlässig arbeitet. In X-Richtung hat der Touchscreen einen Arbeitsbereich von 214,5 mm und in Y-Richtung eine Bereich von 161,0 mm. Diese Werte wurden durch Messungen ermittelt.

Durch Ausprobieren wurden die maximal und minimal ADC-Werte in die jeweilige Richtung ermittelt.

Tabelle 5.1: Empirisch ermittelter Wertebereich der ADC für die äußeren Grenzen des Touchscreens in jeweils horizontaler (x) und vertikaler (y) Richtung.

	Min	Max
x	68	961
y	108	917

Mit diesen Werten lässt sich Arbeitsbereich (in ADC-Werten) des Touchscreen in jede Richtung bestimmen.

$$ADC_{x,len} = 961 - 68 = 893 \tag{5.1}$$

$$ADC_{y,len} = 917 - 108 \tag{5.2}$$

Mit diesen Werten (eq. (5.1) und eq. (5.2)) können im Anschluss die Werte in das metrische System überführt werden und die Auflösung des Touchscreen bestimmt werden. In x-Richtung ergibt sich eine Auflösung von 0,240 $\frac{\text{mm}}{\text{ADC}}$ und in y-Richtung 0,199 $\frac{\text{mm}}{\text{ADC}}$.

Diese unterschiedliche Werte haben den Ursprung, dass die ADC-Werte sich in x-Richtung auf eine größere Distanz verteilen als in y-Richtung.

5.1 Genauigkeit bei konstanten Koordinaten

Bei dieser Untersuchung wurden zwei separate Messungen durchführen. Im ersten Durchlauf wurden die Werte mit dem Medianfilter verarbeitet, bevor sie ausgegeben wurden. Im zweiten Durchlauf wurden die direkten und ungefilterte Werte ausgegeben. In den fig. 5.1 und fig. 5.2 sind die Messdaten der x- und y-Komponenten aufgetragen (siehe Seite fig. 5.1).

Die Auswertung der Messdaten ist in table 5.2 und table 5.3 zu finden. Bei der Auswertung ist zu beachten das es um zwei separate Messreihen handelt. Daher hat auch die ungefilterte Messreihe eine Standardabweichung und Varianz von Null, im Vergleich zur gefilterten Messreihe. Im Normalbetrieb ist der Medianfilter im Programm aktiv, daher haben die Werte der gefilterten Messreihe eine höhere Relevanz. Die Genauigkeit des Touchscreen in beiden Messreihen ist kleiner als die Auflösung, was auf ein akkurat arbeitenden Touchscreen schließen lässt.

Tabelle 5.2: Auswertung der gefilterten Messdaten

	Median		Standardabweichung		Varianz	
Einheit	(ADC)	mm	(ADC)	mm	(ADC)	mm
x-Richtung	499,0	119,861	0,0	0,0	0,0	0,0
y-Richtung	509,999	101,495	0,049	0,010	0,0	0,0

Tabelle 5.3: Auswertung der ungefilterten Messdaten

	Median		Standardabweichung		Varianz	
Einheit	(ADC)	mm	(ADC)	mm	(ADC)	mm
x-Richtung	499,0	119,861	0,0	0,0	0,0	0,0
y-Richtung	510,0	101,496	0,0	0,0	0,0	0,0

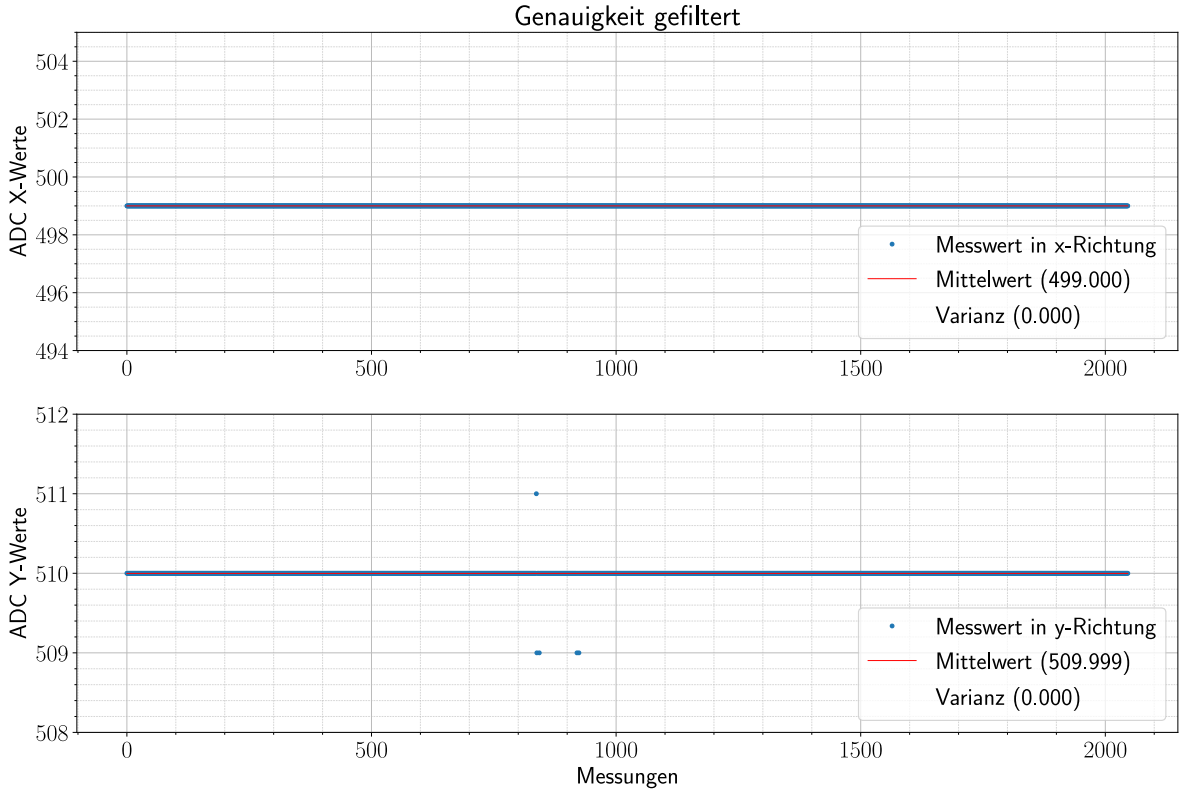


Abbildung 5.1: Darstellung der gefilterten Messreihe

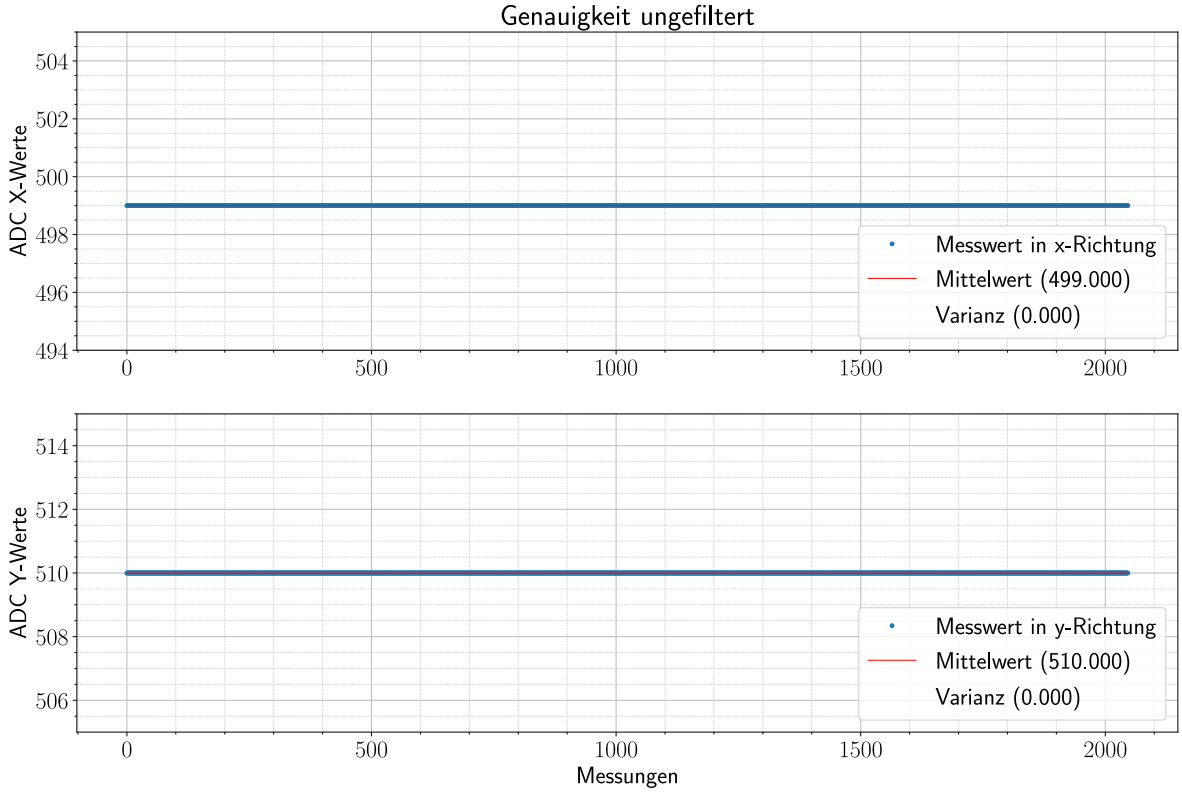


Abbildung 5.2: Darstellung der ungefilterten Messreihe

5.2 Reproduzierbarkeit von Koordinaten

Um die Reproduzierbarkeit von Koordinaten zu untersuchen, wurde die Mitte des Touchscreen mehrmals berührt während die Messdaten aufgezeichnet wurden. Die Auswertung der Messdaten sind in table 5.4 zu sehen. Aus den Werten kann man sagen, dass die Genauigkeit der Auflösung entspricht.

Tabelle 5.4: Auswertung der Reproduzierbarkeit von Koordinaten

	Median		Standardabweichung		Varianz	
Einheit	(ADC)	mm	(ADC)	mm	(ADC)	mm
x-Richtung	510,75	122,683	0,894	0,215	0,8	0,192
y-Richtung	515,858	102,661	1,159	0,231	1,3	0,259

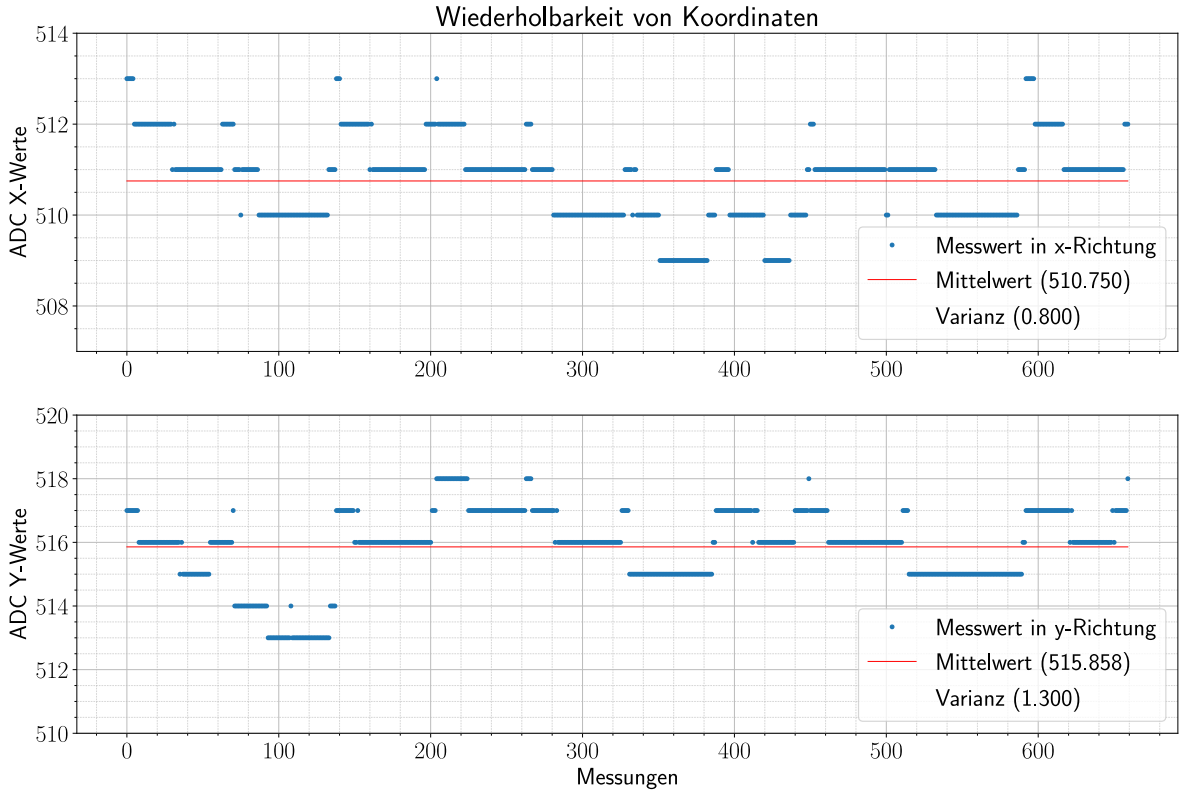


Abbildung 5.3: Darstellung der Reproduzierbarkeit von Koordinaten

5.3 Linearität in x- und y-Richtung

Um eine Aussage über die Linearität des Touchscreens treffen zu können, wurden in x- und y-Richtung, auf dem Touchscreen alle 10 mm eine Markierung gesetzt (siehe fig. 5.6).

Die jeweilige Komponenten wurde anschließend jeweils über die physikalische Strecke in einem Diagramm dargestellt (siehe fig. 5.7 und fig. 5.8). Die Messwerte wurden mittels einer linearen Anpassung gefittet (siehe fig. 5.4 und fig. 5.5).

Fit Statistics

```
# fitting method   = leastsq
# function evals   = 6
# data points      = 22
# variables        = 2
chi-square         = 0.29365924
reduced chi-square = 0.01468296
Akaike info crit   = -90.9603088
Bayesian info crit = -88.7782239
```

Variables

```
slope:     -4.19974387 +/- 0.00106999 (0.03%) (init = -4.198905)
intercept:  961.484867 +/- 0.05144124 (0.01%) (init = 960.8882)
```

Abbildung 5.4: Auswertung der Linearität in x-Richtung

Fit Statistics

```
# fitting method   = leastsq
# function evals   = 6
# data points      = 17
# variables        = 2
chi-square         = 5.94585064
reduced chi-square = 0.39639004
Akaike info crit   = -13.8588356
Bayesian info crit = -12.1924089
```

Variables

```
slope:     -5.03128570 +/- 0.00594634 (0.12%) (init = -5.071253)
intercept:  915.677092 +/- 0.43126811 (0.05%) (init = 921.4577)
```

Abbildung 5.5: Auswertung der Linearität in y-Richtung

Das χ^2 gibt Auskunft darüber in welchem Maß Werte miteinander sich verändern. Je kleiner dieser Wert ist desto eher stimmt die Linearität überein. Bei der linearen Anpassung in x-Richtung wurde ein χ^2 von $0,294$ ermittelt. Für die Linearität in y-Richtung wurde ein χ^2 von $5,946$ ermittelt.

Bei der Untersuchung, der Linearität in y-Richtung, gibt es bei Abstand 40 mm ein Messpunkt der von der Messpunktewolke und der dazugehörigen linearen Anpassung abweicht. Dieser Messpunkt führt zu diesem größeren χ^2 als im Vergleich zur Messreihe in x-Richtung.

Um Abschließend eine Aussage treffen zu können, ob diese Werte im Wertebereich des Datenblatts sind (appendix A.1, page 15), muss der Grenzwert der Chi-Quadrat-Verteilung mit den Werten der Linearen Anpassung verglichen werden. Im Datenblatt wird eine Linearität von 1,5 % garantiert. In der Wertetabelle von [Pap17] gibt es nur Werte für 1 % oder 2,5 %. Der gelistete Wert für zwei Freiheitsgrade und für 1 % liegt bei 7,88. Sowohl das χ^2 in x-Richtung wie auch in y-Richtung ist kleiner diesem Werte. Dies hat zur Folge, dass dieser Touchscreen eine Linearität von unter 1 % aufweist.

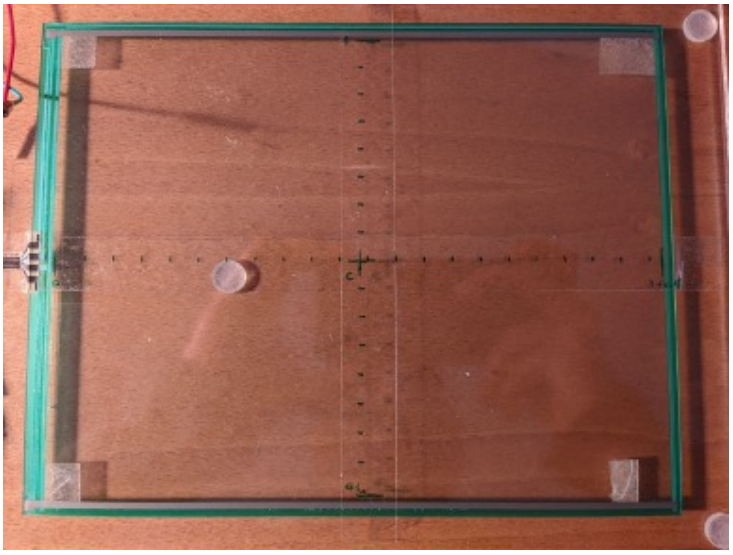


Abbildung 5.6: Messaufbau für Linearität in x- und y-Richtung

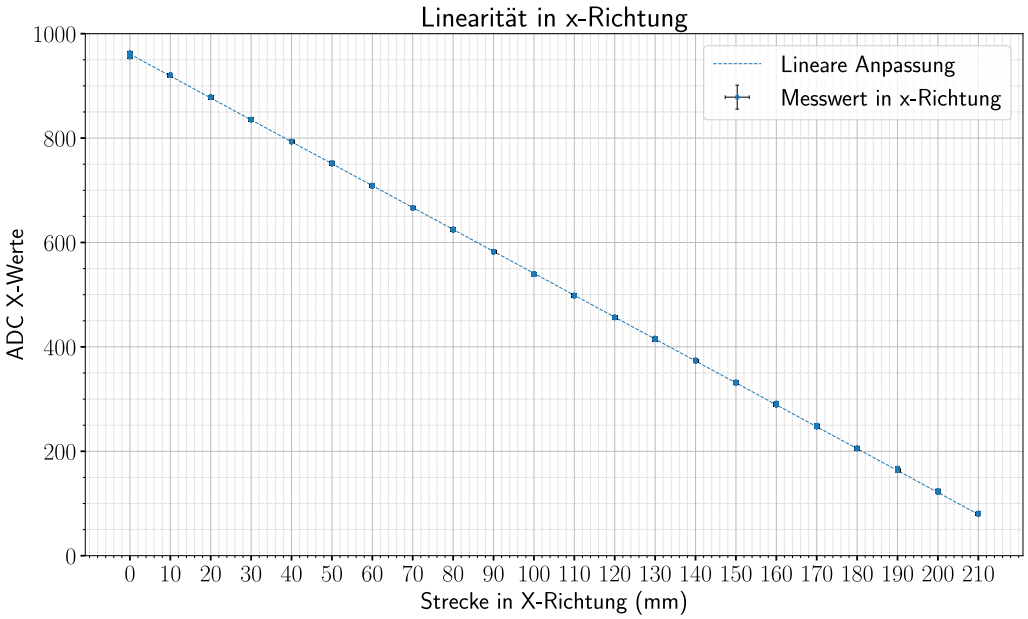


Abbildung 5.7

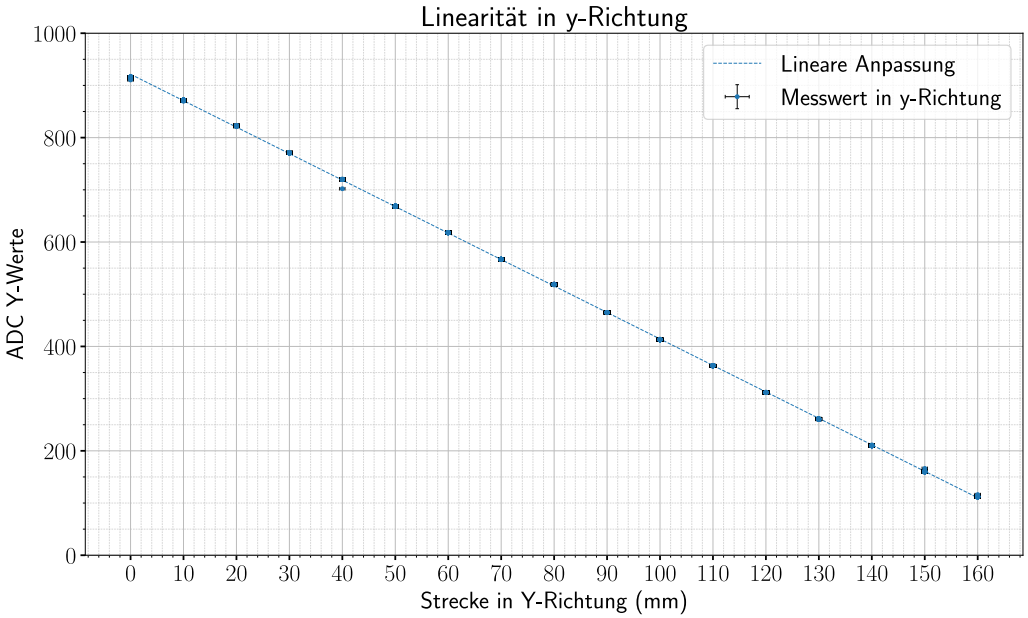


Abbildung 5.8

6 Fazit

6.1 Qualität der Erkenntnisse

Der uns ausgehändigte Touchscreen weißt zum einen eine ausgeprägte Genauigkeit wie auch Linearität auf. Es entspricht den Erwartungen und arbeitet zuverlässig. Vor allem wenn ein Punkt übe längere Zeit bedrückt wird, weißt der Touchscreen eine hohe Genauigkeit auf.

6.2 Verbesserungsvorschläge

A

Anhang

A.1 Touchscreen

FUJITSU Component Touch Panels
Standard 4-Wire series

Fujitsu Resistive Touch Panel Specification

Features

- Superior quality standard 4-wire resistive analog touch panel
- Excellent specification and high quality
  - Anti Newton ring technology
  - High reliability materials
- Pen/finger type
- Transparency - 80% typical
- RoHS compliant

Part Numbers

Part Number	Size	Type
N010-0554-T703A	3.8"	Pen Finger (80%, AG, 21mm FPC)
N010-0554-T241A	4.3"W	Pen Finger (86%, AS, 75mm FPC)
N010-0554-T015A	5.7"	Pen/Finger (120mm FPC)
N010-0554-T009A	5.7"	Pen/Finger (50mm FPC)
N010-0554-T043A	6.4	Pen/Finger (80%, AG, 61mm FPC)
N010-0554-T048A	6.4"	Pen/Finger (86% clear, 120mm FPC)
T010-1301-T320*	7"W	Pen/Finger (82%, AS, 70mm FPC)
N010-0554-T504A	8.4"	Pen/Finger (0.7mm glass, AS 75mm FPC)
N010-0519-T742A	8.4"	Pen/Finger (86%, clear, 120mm FPC)
N010-0554-T511A	8.4"	Pen/Finger (1.1mm glass, AS 120mm FPC)
T010-1201-T930*	10.1"	Pen/Finger (83%, AG, 80mm FPC)
N010-0554-T347A	10.4"	Pen/Finger (75mm FPC)
N010-0554-T352A	10.4"	Pen/Finger (82%, AS, 120mm FPC)
N010-0554-T351A	10.4"	Pen/Finger (86%, AS, 120mm FPC)
N010-0554-T805A	12.1"	Pen/Finger (75mm FPC)
N010-0554-T814A	12.1"	Pen/Finger (82% 120 mm FPC)
N010-0554-T902A	15"	Pen/Finger (61mm FPC)

Notes

- Unless otherwise noted, all PNs are 1.1mm Glass, Pen/Finger operation, 80% Transmissivity, Anti-Glare (AG), RoHS-Compliant.
- AS = Anti-Smudge, aka AFP or Anti-Finger Print
- For drawings please refer to the Documentation tab.
- Full specifications are available - please contact your local Sales Representative, or use the Contact Form, to request.

*: Produced by Transtouch Technology, Inc., a Fujitsu partner company.

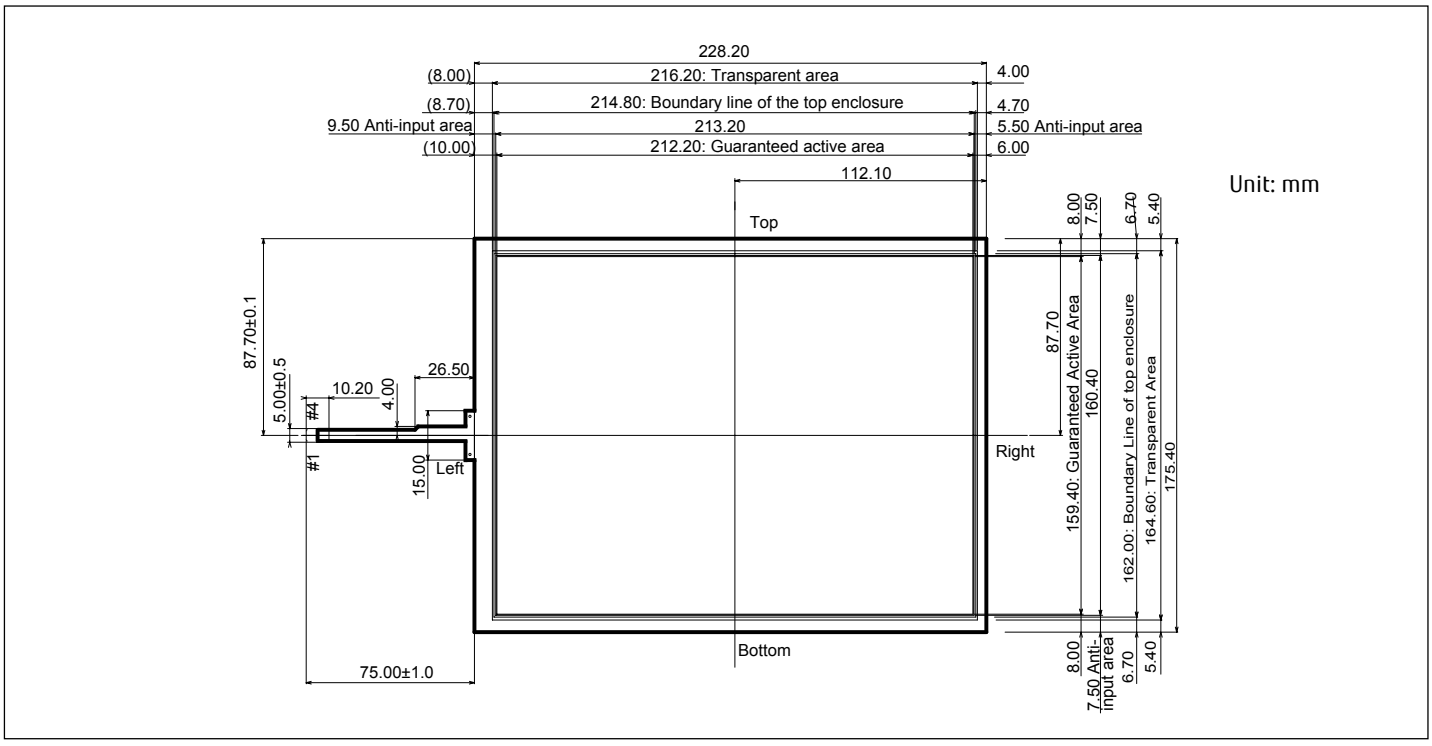
■ Controller Boards

Part Number	Type
NC01850-B070RS	4-wire, RS232
NC01850-B010RS	4-wire, USB
FID-1850-120	4-wire, USB, dual touch

■ Interface Controller Chips

Part Number	Type
NC41120-0036	4-wire, RS232
NC41120-0051	4-wire, USB
FID-1860-005	4-wire, USB, dual touch

■ Dimension example (10.4" shown)



■ Detailed specifications

■ 1.0 Application

This specification applies to the standard series Resistive Touch Panel (Pen/Finger type)

■ 2.0 Additional application

Complete specification document is available upon request.

■ 3.0 Description and block diagram

This panel in combination with a control IC chip or control board is used to transfer the co-ordinate data to the host system.
Please see block diagram on page 1.

■ Electrical

Rated voltage	DC 7V max.
Resistance X axis	300 to 850Ω (at the connector) typical
Resistance Y axis	100 to 600Ω (at the connector) typical
Switch bounce (chattering)	20ms min. when using the silicon rubber measurement tool
Insulation resistance	10MΩ minimum
Dielectric strength	25KV DC for 1 minute
Linearity	1.5% typical

■ Environmental

Operating temperature (*)	-5°C to 60°C
Storage temperature	-20°C to 70°C
Operating humidity	20% to 85% RH with a maximum wet bulb temperature of 38°C
Storage humidity	10% to 90% RH with a maximum wet bulb temperature of 38°C
Chemical resistance	Coating with the following chemicals and storing at room temperature for 2 hours gives no problems. 10% NaCl-water solution, ethyl-acetate, ethyl-alcohol, toluene, methyl-ethyl-ketone
Low air pressure	No issues down to 0.5 x normal air pressure

■ General notes

Touch panels are made of glass, so care must be taken in handling them. Do not stress, pile, bend, lift by the cable or put any stress on the film, for example moving by film face vacuum. In order to clean wring dry a cloth which has been emersed in a natural detergent.
DO NOT use any organic solvent, acid or alkali solution. Watch the edge of the panel when cleaning, again for safety reasons.

Contact

Japan
FUJITSU COMPONENT LIMITED
Shinagawa Seaside Park Tower
12-4, Higashi-shinagawa 4-chome,
Tokyo 140 0002, Japan
Tel: (81-3) 3450-1682
Fax: (81-3) 3474-2385
Email: fcl-contact@cs.jp.fujitsu.com
Web: www.fujitsu.com/jp/group/fcl/en/

North and South America
FUJITSU COMPONENTS AMERICA, INC.
2290 North First Street, Suite 212
San Jose, CA 95131 U.S.A.
Tel: (1-408) 745-4900
Fax: (1-408) 745-4970
Email: components@us.fujitsu.com
Web: http://us.fujitsu.com/components/

Europe
FUJITSU COMPONENTS EUROPE B.V.
Diamantlaan 25
2132 WV Hoofddorp, The Netherlands
Tel: (31-23) 5560910
Fax: (31-23) 5560950
Email: info@fceu.fujitsu.com
Web: www.fujitsu.com/uk/products/devices/components/

Asia Pacific
FUJITSU COMPONENTS ASIA, Ltd.
102E Pasir Panjang Road
#01-01 Citilink Warehouse Complex,
Singapore 118529
Tel: (65) 6375-8560 / Fax: (65) 6273-3021
Email: fcal@sg.fujitsu.com
www.fujitsu.com/sg/products/devices/components/

China
FUJITSU ELECTRONIC COMPONENTS (SHANGHAI) CO., LTD.
Unit 4306, InterContinental Center
100 Yu Tong Road, Shanghai 200070, China
Tel: (86 21) 3253 0998 /Fax: (86 21) 3253 0997
Email: fcal@sg.fujitsu.com
www.fujitsu.com/sg/products/devices/components/

Hong Kong
FUJITSU COMPONENTS HONG KONG Co., Ltd.
Room 06, 28/F, Greenfield Tower, Concordia Plaza, No.1 Science Museum Road,
Tsim Sha Tsui East, Kowloon, Hong Kong
Tel: (852) 2881 8495 Fax: (852) 2894 9512
Email: fcal@sg.fujitsu.com
www.fujitsu.com/sg/products/devices/components/

Korea
FUJITSU COMPONENTS KOREA, LTD.
Alpha Tower #403,
645 Sampyeong-dong,
Bundang-gu, Seongnam-si,
Gyeonggi-do, 13524 Korea
Tel: (82 31) 708-7108
Fax: (82 31) 709-7108
Email: fcal@sg.fujitsu.com
www.fujitsu.com/sg/products/devices/components/

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Fujitsu:
N010-0556-T408 N010-0554-T703 N010-0554-T347 N010-0554-T504 N010-0554-T805 N010-0554-T009 N010-0554-T048 N010-0554-T043 N010-0516-T947 N010-0514-T003 N010-0514-T005 N010-0554-T015 N010-0554-T813 N010-0554-T814 N010-0516-T104 N010-0519-T742 N010-0516-T407 N010-0554-T352 N010-0554-T351 N010-0514-T101 N010-0554-T902 N010-0554-T241

B Firmware

Programmcode B.1: Main

```
#ifndef ARDUINO_H
#include <Arduino.h>
#define ARDUINO_H
#endif
#include "defines.h"
#include "FourWireRTP.h"
#include <Mouse.h>

/*
Port mapping: A0 - A3 -> PF7 - PF4
Pin    Port    Color    Position
-----------------------------
A0     PF7     blk      x_le
A1     PF6     brn      x_ri
A2     PF5     red      y_up
A3     PF4     ylw      y_lo

ToDo:
[x] Improve filtering (?)
    Note: added median filter
[x] Fix touch-release bug
[x] Output decimals to serial
[ ] Add calibration routine (?)
[ ] Manual calibration
[x] Fix dead area at outer most right side
    Note: fixing touch-release bug fixed this one as well
*/

int xvals[OVERSAMPLING] {0};
int yvals[OVERSAMPLING] {0};
// int *p;
float xval;
float yval;

void setup() {
  cli();
  // Setting unused pins as Input and activate pull-ups
  DDRB = 0x00;
  PORTB = 0xFF;
  DDRC = 0x80;                                    // LED_BUILTIN (PC7) as output
  PORTC = 0x7F;                                   // LED_BUILTIN off
  DDRD = 0x00;
  PORTD = 0xFF;
  DDRE = 0x00;
  PORTE = 0xFF;
  // Panel is connected to PORTF pins PF7:4
```

```
47    DDRF = 0x80;                                          // All pins as
          input
      PORTF = 0x20;                                         // All pins no
          pull-up / LOW
49
      // Pre-configuring ADC
51    ADMUX |= (1<<REFS0) | (1<<MUX2) | (1<<MUX0);  // Set VRef to Vcc and
          connect ADC5 to ADC (p. 313)
      // ADC enable and ADC-clock prescaler to 128. CPU-Clock / ADC-prescaler =
           ADC-Clock.
53    // At 10bit resolution must not be higher than 200kHz.
      // ADPS2:0 set to 1 prescales by 128 giving an ADC-clock of 125kHz (8us
          per cycle).
55    // Sampling + conversion takes up 1.5+13 cycles -> 12+104us
      // See p. 300 ff
57    ADCSRA |= (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0);

59    // Configuring Timer0
      TCCR0B |= (1<<CS01) | (1<<CS00);                      // Prescaler to
           64 -> Translates to 250kHz or 4us/cycle
61    OCR0A = 24;                                           // ADC Sample-
          and-hold 1.5cycles -> 12us. Thus, TOCM trigger at 24

63    /*
      Configuring interrupts (p. 89)
65    ISCn1     ISCn0     Description
      0         0         The low level of INTn generates an interrupt request
67    0         1         Any edge of INTn generates asynchronously an interrupt
          request
      1         0         The falling edge of INTn generates asynchronously an
          interrupt request
69    1         1         The rising edge of INTn generates asynchronously an
          interrupt request
      */
71    EICRA |= (1<<ISC00);
      EIMSK |= (1<<INT0);                                   // Enable
          external interrupt on INTO
73
      sei();
75    Serial.begin(115200);
      Mouse.begin();
77
      delay(300);
79  }


81

83  void loop() {
      // delay(1);
85
      if (isFingered()) {
87      for (int i = 0; i < OVERSAMPLING; i++) {
          xvals[i] = readX();
89      }
        // xval = readX();
91      if (isFingered()) {
          for (int i = 0; i < OVERSAMPLING; i++) {
93          yvals[i] = readY();
          }
```

```
95          xval = doSomeMedianFiltering(xvals, OVERSAMPLING, CLAMP);
            yval = doSomeMedianFiltering(yvals, OVERSAMPLING, CLAMP);
97
            // Mouse control values are only send if MOUSE_EN (PIND6) is pulled
                LOW,
99          // else positional vales are sent over serial
            if (!(PIND & (1<<MOUSE_EN))) {
101           xval = map(xval, XMIN, XMAX, -MOUSE_SPEED, MOUSE_SPEED);
              yval = map(yval, YMIN, YMAX, MOUSE_SPEED, -MOUSE_SPEED);
103           Mouse.move(xval, yval, 0);
              // Serial.print(xval);Serial.print("\t");Serial.println(yval);
105           delay(MOUSE_DELAY);
            } else {
107           xval = map(xval, XMIN, XMAX+1.0, 5*WIDTH, -5*WIDTH+1)/10.0;
              yval = map(yval, YMIN, YMAX+1.0, -5*HEIGHT, 5*HEIGHT+1)/10.0;
109           Serial.print(xval,1);Serial.print("\t");Serial.println(yval,1);
            }
111       }
      }
113 }

115 ISR(INT0_vect){
      // Let LED_L indicate if mouse control is enabled. ON if turned on, OFF
          otherwise
117   if (PIND & (1<<MOUSE_EN)){
        PORTC &= ~(1<<LED_L);                                   // LED_L off
119   } else {
        PORTC |= (1<<LED_L);                                    // LED_L on
121   }
    }
123
    ISR(TIMER0_COMPA_vect) {
125   // setTOCM(false);
      DDRF |= (1<<DDF5) | (1<<DDF4);                            // PF5 and PF4
          as output
127   PORTF |= (1<<PORTF5) | (1<<PORTF4);                       // PF5 and PF4
          HIGH
      // DDRF = 0x30;
129   // PORTF = 0x30;
    }
```

Programmcode B.2: Helper Functions

```
#include <Arduino.h>
#include "helper_functions.h"

void setTOCM(bool set){
        cli();
        if (set){
                TIMSK0 |= (1<<OCIE0A);
                TCNT0 = 0;
        } else {
                TIMSK0 &= ~(1<<OCIE0A);
        }
        sei();
}

bool isFingered(void){
        /*
        PF7   x_le     LOW
        PF6   x_ri     Hi Z
        PF5   y_up     Hi Z
        PF4   y_lo     ADC
        */

        DDRF = 0x80;
                                                // Set PF7 as output, others as
            input
        PORTF = 0x10;
                                                // Pull-up at ADC5 and set PF7
            LOW
        ADMUX = 0x44;
                                                // Connect ADC4
        // setTOCM(true);
        int check = getADC();
        if (check < 100) {
                return true;
        } else {
                // Serial.print(check);Serial.println("\t No");
                return false;
        }
}

uint16_t getADC(){
        setTOCM(true);
                                                // Enable Timer Output Compare
            Match interrupt
        ADCSRA |= (1<<ADSC) | (1<<ADEN);
                                // Enable ADC and start conversion
        while ((ADCSRA & (1<<ADSC))){
                                // Wait for Conversion to complete (about 13 ADC
             cycles/104us)
                // do nothing
        }
        uint16_t val = ADCL;
                                        // Read ADCL ...
        val |= (ADCH<<8);
                                                // and add ADCH left shifted by
            8 bit
        return val;
```

```
}

int readX() {
        /*
        Reading X               | Reading Y
        ------------------------------------------------
        ADC7   x_le     HIGH    | ADC7   x_le     ADC
        ADC6   x_ri     LOW     | ADC6   x_ri     Hi Z
        ADC5   y_up     ADC     | ADC5   y_up     HIGH
        ADC4   y_lo     Hi Z    | ADC4   y_lo     LOW
        */

        setTOCM(false);
        // Setting up pin modes
        DDRF = 0xC0;
                                                // Setting PF7 and PF6 as output
        PORTF = 0x80;
                                                // Setting PF7 HIGH
        ADMUX = 0x45;
                                                // Making sure ADC5 is connected
            to ADC
        // Enable ADC and start conversion
        // ADCSRA |= (1<<ADSC) | (1<<ADEN);
        return getADC();
}

int readY() {
        /*
        Reading X               | Reading Y
        ------------------------------------------------
        ADC7   x_le     HIGH    | ADC7   x_le     ADC
        ADC6   x_ri     LOW     | ADC6   x_ri     Hi Z
        ADC5   y_up     ADC     | ADC5   y_up     HIGH
        ADC4   y_lo     Hi Z    | ADC4   y_lo     LOW
        */

        setTOCM(false);
        // Setting up pin modes
        DDRF = 0x30;
                                                // Setting PF5 and PF4 as output
        PORTF = 0x20;
                                                // Setting PF5 HIGH
        ADMUX = 0x47;
                                                // Connecting ADC7 to ADC
        // Enable ADC and start conversion
        // ADCSRA |= (1<<ADSC) | (1<<ADEN);
        return getADC();
}

// float doSomeAveraging(float vals[]) {
//   float avrg = 0;
//   for (int i = 0; i < OVERSAMPLING; i++) {
//     avrg += vals[i];
//   }
//   return avrg / OVERSAMPLING;
// }

float doSomeMedianFiltering(int *p, int n, int clamp) {
  /*
```

```
    Implementation of median filter. All values in the range
98
      <value at center of array>-clamp < value < <value at center of array>+
          clamp
100
    will be averaged to the final value.
102 */
    int m = 0;
104 int sum = 0;
    for (int i = 0; i < n; i++) {
106   if ((*p+n/2 < (*p+n/2)+clamp) | (*p+n/2 > (*p+n/2)-clamp)) {
        sum += *p;
108     m++;
      }
110   *p++;
    }
112 return sum / (float)m;
}
```

Programmcode B.3: Defines

```
1  #pragma once

3  // Range of raw ADC values accross the touch screen area
   #define XMIN 69.0
5  #define XMAX 954.0
   #define YMIN 102.0
7  #define YMAX 913.0

9  // Size of the arrays holding ADC readings
   #define OVERSAMPLING 40
              // Time to get one xy-pair: ((OVERSAMPLING*2)+2)/ADC_freq
11 #define CLAMP 5

13 // Physical dimensions of the sensitive area of the touch screen (in mm)
   #define WIDTH 212.2
15 #define HEIGHT 159.4

17 // Offsets to place the coordinate origin at the center of the sensitive
       area
   #define VOFFSET -14.0
19 #define HOFFSET 19.0

21 // Mouse stuff
   #define MOUSE_EN PIND0
23 #define MOUSE_SPEED 20
   #define MOUSE_DELAY 5
25
   // Check touchy
27 #define FINGERED_UP !(PINF & (1<<PINF5))
   #define FINGERED_LO !(PINF & (1<<PINF4))
29
   // Bit position of the LED_L
31 #define LED_L 7
```

Literatur

[Mil] Nelson Miller. 4-Wire vs 5-Wire Resistive Touchscreens: What's the Difference? URL: https://www.nelson-miller.com/4-wire-vs-5-wire-resistive-touchscreens-whats-difference/ (besucht am 08. 02. 2022).

[Pap17] Lothar Papula. Mathematische Formelsammlung. 12. Auflage. München: Springer Vieweg, 2017. ISBN: 978-3-658-16194-1.

Abbildungsverzeichnis

Tabellenverzeichnis

Programmcodeverzeichnis