# Konzept und Prototyp für ein taktiles Sensorarray

**Nikolas Slottke**[1] **Hendrik Udo Linne**[2]
{7slottke, 7linne}@informatik.uni-hamburg.de

Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Department Informatik
**Technische Aspekte Multimodaler Systeme**

26. April 2011

# Gliederung

# Motivation

In der Robotik ist es eine der Herausforderungen Objekte mit Hilfe von Robotern zu manipulieren. Im speziellen Fall sogar mit mechanischen Händen. Aus diesem Grund ist 2009 das Handle-Project gestartet worden. Mehrere europäische Universitäten nehmen bis 2013 daran teil.
Die Motivation der Bachelorarbeit ist das Bestreben die menschliche Benutzung und vor Allem den menschlichen Tastsinn nachzuvollziehen und für mechanische Hände nachzuempfinden. Neben der Möglichkeit den Druck auf ein Objekt zu empfinden ist der Mensch in der Lage, die Position seiner Hand auf dem Objekt zu erkennen.

# Zielsetzung

Genau diese Positionserkennung der Hand auf dem Objekt ist Bestandteil der vorliegenden Bachelorarbeit.
Die Zielsetzung ist es einen taktilen Sensor zu konstruieren, der in der Lage ist eine Positionsbestimmung zwischen Hand und Objekt zu realisieren.

# Konzept

Der Datenhandschuh und ein Objekt sollen mit einem Sensorfeld ausgestattet werden.

- Bestimmung der genauen Position eines Berührungspunktes auf Handschuh und Objekt
- Verbindungen von Hand und Objekt werden durch elektrische Kontakte realisiert
- Sensoren auf Handschuh und Objekt sollen Kontakte bzw. Kontaktflächen sein
- Mikrocontroller von Atmel werden mit ihren Ein- und Ausgängen an die Kontakte angeschlossen

# Konzept

Der Datenhandschuh und ein Objekt sollen mit einem Sensorfeld ausgestattet werden.

- Bestimmung der genauen Position eines Berührungspunktes auf Handschuh und Objekt
- Verbindungen von Hand und Objekt werden durch elektrische Kontakte realisiert
- Sensoren auf Handschuh und Objekt sollen Kontakte bzw. Kontaktflächen sein
- Mikrocontroller von Atmel werden mit ihren Ein- und Ausgängen an die Kontakte angeschlossen

# Konzept

Der Datenhandschuh und ein Objekt sollen mit einem Sensorfeld ausgestattet werden.

- Bestimmung der genauen Position eines Berührungspunktes auf Handschuh und Objekt
- Verbindungen von Hand und Objekt werden durch elektrische Kontakte realisiert
- Sensoren auf Handschuh und Objekt sollen Kontakte bzw. Kontaktflächen sein
- Mikrocontroller von Atmel werden mit ihren Ein- und Ausgängen an die Kontakte angeschlossen

# Konzept

Der Datenhandschuh und ein Objekt sollen mit einem Sensorfeld ausgestattet werden.

- Bestimmung der genauen Position eines Berührungspunktes auf Handschuh und Objekt
- Verbindungen von Hand und Objekt werden durch elektrische Kontakte realisiert
- Sensoren auf Handschuh und Objekt sollen Kontakte bzw. Kontaktflächen sein
- Mikrocontroller von Atmel werden mit ihren Ein- und Ausgängen an die Kontakte angeschlossen

# Konzept

- Kontaktflächen des Handschuhs als Ausgang
- Kontaktflächen des Objektes als Eingang
- Nacheinander soll je 1 Pin der Hand auf High geschaltet werden, danach Abfragen vom Zustand der Objektsensoren
- Genaue Rekonstruktion welcher Kontakt mit welchem verbunden ist möglich

# Konzept

- Kontaktflächen des Handschuhs als Ausgang
- Kontaktflächen des Objektes als Eingang
- Nacheinander soll je 1 Pin der Hand auf High geschaltet werden, danach Abfragen vom Zustand der Objektsensoren
- Genaue Rekonstruktion welcher Kontakt mit welchem verbunden ist möglich

# Konzept

- Kontaktflächen des Handschuhs als Ausgang
- Kontaktflächen des Objektes als Eingang
- Nacheinander soll je 1 Pin der Hand auf High geschaltet werden, danach Abfragen vom Zustand der Objektsensoren
- Genaue Rekonstruktion welcher Kontakt mit welchem verbunden ist möglich

# Konzept

- Kontaktflächen des Handschuhs als Ausgang
- Kontaktflächen des Objektes als Eingang
- Nacheinander soll je 1 Pin der Hand auf High geschaltet werden, danach Abfragen vom Zustand der Objektsensoren
- Genaue Rekonstruktion welcher Kontakt mit welchem verbunden ist möglich

# Konzept

Gesteuert werden soll das ganze von einem Kommunikationscontroller.

- Steuerung von Handcontrollern
- Steuerung von Objektcontrollern
- Verwalten der Zustandsdaten sowohl von Hand als auch von Objekt
- Teilweise Fehlererkennung
- Übermittlung der Daten an den PC

# Konzept

Gesteuert werden soll das ganze von einem Kommunikationscontroller.

- Steuerung von Handcontrollern
- Steuerung von Objektcontrollern
- Verwalten der Zustandsdaten sowohl von Hand als auch von Objekt
- Teilweise Fehlererkennung
- Übermittlung der Daten an den PC

# Konzept

Gesteuert werden soll das ganze von einem Kommunikationscontroller.

- Steuerung von Handcontrollern
- Steuerung von Objektcontrollern
- Verwalten der Zustandsdaten sowohl von Hand als auch von Objekt
- Teilweise Fehlererkennung
- Übermittlung der Daten an den PC

# Konzept

Gesteuert werden soll das ganze von einem Kommunikationscontroller.

- Steuerung von Handcontrollern
- Steuerung von Objektcontrollern
- Verwalten der Zustandsdaten sowohl von Hand als auch von Objekt
- Teilweise Fehlererkennung
- Übermittlung der Daten an den PC

# Konzept

Gesteuert werden soll das ganze von einem Kommunikationscontroller.

- Steuerung von Handcontrollern
- Steuerung von Objektcontrollern
- Verwalten der Zustandsdaten sowohl von Hand als auch von Objekt
- Teilweise Fehlererkennung
- Übermittlung der Daten an den PC

# Konzept

Kommunikation mit dem PC und anderen Controllern durch:

- TWI-Bus (two wire interface) mit den anderen Controllern
- Jeder Controller ist durch eine eigene Adresse ansprechbar
- Kommunikation mit dem PC via USART am Mikrocontroller zur seriellen Schnittstelle des PCs
- Befehle vom PC an den Mikrocontroller senden (wurde von uns getestet, aber im jetzigen Programm nicht implementiert bzw. auskommentiert)

# Konzept

Kommunikation mit dem PC und anderen Controllern durch:

- TWI-Bus (two wire interface) mit den anderen Controllern
- Jeder Controller ist durch eine eigene Adresse ansprechbar
- Kommunikation mit dem PC via USART am Mikrocontroller zur seriellen Schnittstelle des PCs
- Befehle vom PC an den Mikrocontroller senden (wurde von uns getestet, aber im jetzigen Programm nicht implementiert bzw. auskommentiert)

# Konzept

Kommunikation mit dem PC und anderen Controllern durch:

- TWI-Bus (two wire interface) mit den anderen Controllern
- Jeder Controller ist durch eine eigene Adresse ansprechbar
- Kommunikation mit dem PC via USART am Mikrocontroller zur seriellen Schnittstelle des PCs
- Befehle vom PC an den Mikrocontroller senden (wurde von uns getestet, aber im jetzigen Programm nicht implementiert bzw. auskommentiert)

# Konzept

Kommunikation mit dem PC und anderen Controllern durch:

- TWI-Bus (two wire interface) mit den anderen Controllern
- Jeder Controller ist durch eine eigene Adresse ansprechbar
- Kommunikation mit dem PC via USART am Mikrocontroller zur seriellen Schnittstelle des PCs
- Befehle vom PC an den Mikrocontroller senden (wurde von uns getestet, aber im jetzigen Programm nicht implementiert bzw. auskommentiert)

# Atmel Mikrocontroller

Es werden Atmel Mikrocontroller vom Typ ATmega644 verwendet. Die Programmierung erfolgt mit C in der Eclipse IDE mit Hilfe vom AVR-Eclipse Plugin und der Atmel AVR-Toolchain.

- 8MHz interner Takt
- 4 Ports a 8 Pins (32 Ein/Ausgänge)
- USART und TWI Einheiten integriert

# Atmel Mikrocontroller

Es werden Atmel Mikrocontroller vom Typ ATmega644 verwendet. Die Programmierung erfolgt mit C in der Eclipse IDE mit Hilfe vom AVR-Eclipse Plugin und der Atmel AVR-Toolchain.

- 8MHz interner Takt
- 4 Ports a 8 Pins (32 Ein/Ausgänge)
- USART und TWI Einheiten integriert

# Atmel Mikrocontroller

Es werden Atmel Mikrocontroller vom Typ ATmega644 verwendet. Die Programmierung erfolgt mit C in der Eclipse IDE mit Hilfe vom AVR-Eclipse Plugin und der Atmel AVR-Toolchain.

- 8MHz interner Takt
- 4 Ports a 8 Pins (32 Ein/Ausgänge)
- USART und TWI Einheiten integriert

# Kommunikationscontroller

Die main-Methode beinhaltet den Programmablauf:

- Initialisieren des USART (Baudrate 38400bps, 8 Datenbits, 2 Stopbits, keine Parität)
- Initialisieren der Variablen (Objektzustände, Adressen, Handzustand, Zyklen, ...)
- Interrupts löschen, 1.5 Sekunden warten
- Ready-Nachricht an den PC senden
- TWI-Bus initialisieren

# Kommunikationscontroller

Die main-Methode beinhaltet den Programmablauf:

- Initialisieren des USART (Baudrate 38400bps, 8 Datenbits, 2 Stopbits, keine Parität)
- Initialisieren der Variablen (Objektzustände, Adressen, Handzustand, Zyklen, ...)
- Interrupts löschen, 1.5 Sekunden warten
- Ready-Nachricht an den PC senden
- TWI-Bus initialisieren

# Kommunikationscontroller

Die main-Methode beinhaltet den Programmablauf:

- Initialisieren des USART (Baudrate 38400bps, 8 Datenbits, 2 Stopbits, keine Parität)
- Initialisieren der Variablen (Objektzustände, Adressen, Handzustand, Zyklen, ...)
- Interrupts löschen, 1.5 Sekunden warten
- Ready-Nachricht an den PC senden
- TWI-Bus initialisieren

# Kommunikationscontroller

Die main-Methode beinhaltet den Programmablauf:

- Initialisieren des USART (Baudrate 38400bps, 8 Datenbits, 2 Stopbits, keine Parität)
- Initialisieren der Variablen (Objektzustände, Adressen, Handzustand, Zyklen, ...)
- Interrupts löschen, 1.5 Sekunden warten
- Ready-Nachricht an den PC senden
- TWI-Bus initialisieren

# Kommunikationscontroller

Die main-Methode beinhaltet den Programmablauf:

- Initialisieren des USART (Baudrate 38400bps, 8 Datenbits, 2 Stopbits, keine Parität)
- Initialisieren der Variablen (Objektzustände, Adressen, Handzustand, Zyklen, ...)
- Interrupts löschen, 1.5 Sekunden warten
- Ready-Nachricht an den PC senden
- TWI-Bus initialisieren

# Kommunikationscontroller

Daraufhin folgt eine while(1)-Schleife:

- Anzahl der Zyklen erhöhen
- for-Schleife mit i von 1 bis 30 (welcher Pin auf High geschaltet wird)
- Pin i des Handcontrollers wird auf High geschaltet
- Rückmeldung der Hand mit i vergleichen, falls unterschiedlich Fehlermeldung ausgeben
- Objekt nach Portstatus fragen
- Portstatus empfangen und in Variablen speichern

# Kommunikationscontroller

Daraufhin folgt eine while(1)-Schleife:

- Anzahl der Zyklen erhöhen
- for-Schleife mit i von 1 bis 30 (welcher Pin auf High geschaltet wird)
- Pin i des Handcontrollers wird auf High geschaltet
- Rückmeldung der Hand mit i vergleichen, falls unterschiedlich Fehlermeldung ausgeben
- Objekt nach Portstatus fragen
- Portstatus empfangen und in Variablen speichern

# Kommunikationscontroller

Daraufhin folgt eine while(1)-Schleife:

- Anzahl der Zyklen erhöhen
- for-Schleife mit i von 1 bis 30 (welcher Pin auf High geschaltet wird)
- Pin i des Handcontrollers wird auf High geschaltet
- Rückmeldung der Hand mit i vergleichen, falls unterschiedlich Fehlermeldung ausgeben
- Objekt nach Portstatus fragen
- Portstatus empfangen und in Variablen speichern

# Kommunikationscontroller

Daraufhin folgt eine while(1)-Schleife:

- Anzahl der Zyklen erhöhen
- for-Schleife mit i von 1 bis 30 (welcher Pin auf High geschaltet wird)
- Pin i des Handcontrollers wird auf High geschaltet
- Rückmeldung der Hand mit i vergleichen, falls unterschiedlich Fehlermeldung ausgeben
- Objekt nach Portstatus fragen
- Portstatus empfangen und in Variablen speichern

# Kommunikationscontroller

Daraufhin folgt eine while(1)-Schleife:

- Anzahl der Zyklen erhöhen
- for-Schleife mit i von 1 bis 30 (welcher Pin auf High geschaltet wird)
- Pin i des Handcontrollers wird auf High geschaltet
- Rückmeldung der Hand mit i vergleichen, falls unterschiedlich Fehlermeldung ausgeben
- Objekt nach Portstatus fragen
- Portstatus empfangen und in Variablen speichern

# Kommunikationscontroller

Daraufhin folgt eine while(1)-Schleife:

- Anzahl der Zyklen erhöhen
- for-Schleife mit i von 1 bis 30 (welcher Pin auf High geschaltet wird)
- Pin i des Handcontrollers wird auf High geschaltet
- Rückmeldung der Hand mit i vergleichen, falls unterschiedlich Fehlermeldung ausgeben
- Objekt nach Portstatus fragen
- Portstatus empfangen und in Variablen speichern

# Kommunikationscontroller

- Falls eine Verbindung besteht (der Portstatus ist ungleich 0), Pin i mit Präfix 'S' über USART senden, dann Portzustand als Hex-Wert mit zugehörigem Portbuchstaben als Präfix senden (A, B, C, D)
- Wenn for-Schleife durchlaufen wurde, Zyklus mit Präfix 'N' senden
- while-Schleife erneut beginnen mit dem nächsten Zyklus

# Kommunikationscontroller

- Falls eine Verbindung besteht (der Portstatus ist ungleich 0), Pin i mit Präfix 'S' über USART senden, dann Portzustand als Hex-Wert mit zugehörigem Portbuchstaben als Präfix senden (A, B, C, D)
- Wenn for-Schleife durchlaufen wurde, Zyklus mit Präfix 'N' senden
- while-Schleife erneut beginnen mit dem nächsten Zyklus

# Kommunikationscontroller

- Falls eine Verbindung besteht (der Portstatus ist ungleich 0), Pin i mit Präfix 'S' über USART senden, dann Portzustand als Hex-Wert mit zugehörigem Portbuchstaben als Präfix senden (A, B, C, D)
- Wenn for-Schleife durchlaufen wurde, Zyklus mit Präfix 'N' senden
- while-Schleife erneut beginnen mit dem nächsten Zyklus

# Kommunikationscontroller

Die vom PC empfangenen Daten sehen beispielsweise so aus:

...
S18
A08
S19
D04
N41

- Bestehende Verbindungen wären: HandPin18 mit ObjektPin4
- HandPin19 mit ObjektPin25
- Pinnummern: Port A (1-8), B (9-16), C (17-22), D (23-30)

# Kommunikationscontroller

Die vom PC empfangenen Daten sehen beispielsweise so aus:

...
S18
A08
S19
D04
N41

- Bestehende Verbindungen wären: HandPin18 mit ObjektPin4
- HandPin19 mit ObjektPin25
- Pinnummern: Port A (1-8), B (9-16), C (17-22), D (23-30)

# Kommunikationscontroller

Die vom PC empfangenen Daten sehen beispielsweise so aus:

...
S18
A08
S19
D04
N41

- Bestehende Verbindungen wären: HandPin18 mit ObjektPin4
- HandPin19 mit ObjektPin25
- Pinnummern: Port A (1-8), B (9-16), C (17-22), D (23-30)

# Kommunikationscontroller

Portzustand als zweistelliger Hexadezimalwert: Beispiel 'FF':

- Der Binärwert wäre '11111111'
- Hier hätten alle 8 Pins des Ports eine Verbindung

Beispiel '08':

- Der Binärwert wäre '00001000'
- Hier hätte Pin 4 des Ports eine Verbindung
- Rechts im Binärwort ist Pin 1, Links Pin 8

# Kommunikationscontroller

Portzustand als zweistelliger Hexadezimalwert: Beispiel 'FF':

- Der Binärwert wäre '11111111'
- Hier hätten alle 8 Pins des Ports eine Verbindung

Beispiel '08':

- Der Binärwert wäre '00001000'
- Hier hätte Pin 4 des Ports eine Verbindung
- Rechts im Binärwort ist Pin 1, Links Pin 8

# Kommunikationscontroller

Portzustand als zweistelliger Hexadezimalwert: Beispiel 'FF':

- Der Binärwert wäre '11111111'
- Hier hätten alle 8 Pins des Ports eine Verbindung

Beispiel '08':

- Der Binärwert wäre '00001000'
- Hier hätte Pin 4 des Ports eine Verbindung
- Rechts im Binärwort ist Pin 1, Links Pin 8

# Kommunikationscontroller

Portzustand als zweistelliger Hexadezimalwert: Beispiel 'FF':

- Der Binärwert wäre '11111111'
- Hier hätten alle 8 Pins des Ports eine Verbindung

Beispiel '08':

- Der Binärwert wäre '00001000'
- Hier hätte Pin 4 des Ports eine Verbindung
- Rechts im Binärwort ist Pin 1, Links Pin 8

# Kommunikationscontroller

Portzustand als zweistelliger Hexadezimalwert: Beispiel 'FF':

- Der Binärwert wäre '11111111'
- Hier hätten alle 8 Pins des Ports eine Verbindung

Beispiel '08':

- Der Binärwert wäre '00001000'
- Hier hätte Pin 4 des Ports eine Verbindung
- Rechts im Binärwort ist Pin 1, Links Pin 8

# Objektcontroller

Die main-Methode beinhaltet den Programmablauf:

- Alle verfügbaren Pins als Eingang schalten
- Interrupts löschen, 0.5 Sekunden warten
- TWI-Bus initialisieren
- Variablen initialisieren

# Objektcontroller

Die main-Methode beinhaltet den Programmablauf:

- Alle verfügbaren Pins als Eingang schalten
- Interrupts löschen, 0.5 Sekunden warten
- TWI-Bus initialisieren
- Variablen initialisieren

# Objektcontroller

Die main-Methode beinhaltet den Programmablauf:

- Alle verfügbaren Pins als Eingang schalten
- Interrupts löschen, 0.5 Sekunden warten
- TWI-Bus initialisieren
- Variablen initialisieren

# Objektcontroller

Die main-Methode beinhaltet den Programmablauf:

- Alle verfügbaren Pins als Eingang schalten
- Interrupts löschen, 0.5 Sekunden warten
- TWI-Bus initialisieren
- Variablen initialisieren

# Objektcontroller

while(1)-Schleife beginnt:

- TWI-Bus wird nach Nachrichten an den Controller abgefragt
- Wenn der Kommunikationscontroller Lesen erwartet, wird der Status ausgelesen und in den Variablen gespeichert
- Wenn Schreiben erwartet wird, werden die Daten in den Variablen an den Kommunikationscontroller gesendet

# Objektcontroller

while(1)-Schleife beginnt:

- TWI-Bus wird nach Nachrichten an den Controller abgefragt
- Wenn der Kommunikationscontroller Lesen erwartet, wird der Status ausgelesen und in den Variablen gespeichert
- Wenn Schreiben erwartet wird, werden die Daten in den Variablen an den Kommunikationscontroller gesendet

# Objektcontroller

while(1)-Schleife beginnt:

- TWI-Bus wird nach Nachrichten an den Controller abgefragt
- Wenn der Kommunikationscontroller Lesen erwartet, wird der Status ausgelesen und in den Variablen gespeichert
- Wenn Schreiben erwartet wird, werden die Daten in den Variablen an den Kommunikationscontroller gesendet

# Handcontroller

Die main-Methode beinhaltet den Programmablauf:

- Alle verfügbaren Pins als Ausgang schalten
- Interrupts löschen, 0.5 Sekunden warten
- TWI-Bus initialisieren
- Variablen initialisieren

# Handcontroller

Die main-Methode beinhaltet den Programmablauf:

- Alle verfügbaren Pins als Ausgang schalten
- Interrupts löschen, 0.5 Sekunden warten
- TWI-Bus initialisieren
- Variablen initialisieren

# Handcontroller

Die main-Methode beinhaltet den Programmablauf:

- Alle verfügbaren Pins als Ausgang schalten
- Interrupts löschen, 0.5 Sekunden warten
- TWI-Bus initialisieren
- Variablen initialisieren

# Handcontroller

Die main-Methode beinhaltet den Programmablauf:

- Alle verfügbaren Pins als Ausgang schalten
- Interrupts löschen, 0.5 Sekunden warten
- TWI-Bus initialisieren
- Variablen initialisieren

# Handcontroller

while(1)-Schleife beginnt:

- TWI-Bus wird nach Nachrichten an den Controller abgefragt
- Der zu setzende Pin wird vom Kommunikationscontroller empfangen und gesetzt
- Daraufhin wird der Zustand zurückgesendet

# Handcontroller

while(1)-Schleife beginnt:

- TWI-Bus wird nach Nachrichten an den Controller abgefragt
- Der zu setzende Pin wird vom Kommunikationscontroller empfangen und gesetzt
- Daraufhin wird der Zustand zurückgesendet

# Handcontroller

while(1)-Schleife beginnt:

- TWI-Bus wird nach Nachrichten an den Controller abgefragt
- Der zu setzende Pin wird vom Kommunikationscontroller empfangen und gesetzt
- Daraufhin wird der Zustand zurückgesendet

# Sensor Array Application

Das Programm zum Empfangen und Auswerten der Daten des Sensorarrays wurde mit Microsoft Visual Studio 2010 in C# geschrieben. Es hat folgende Aufgaben/Funktionen:

- Verbinden mit unterschiedlichen COM-Ports
- Daten empfangen, auswerten
- Kontaktmatrizen und Verbindungen grafisch darstellen
- Datenstrom der seriellen Schnittstelle anzeigen
- Einzelne Verbindungen als Text anzeigen
- Anzahl der Zyklen und Buffer anzeigen

# Sensor Array Application

Das Programm zum Empfangen und Auswerten der Daten des Sensorarrays wurde mit Microsoft Visual Studio 2010 in C# geschrieben. Es hat folgende Aufgaben/Funktionen:

- Verbinden mit unterschiedlichen COM-Ports
- Daten empfangen, auswerten
- Kontaktmatrizen und Verbindungen grafisch darstellen
- Datenstrom der seriellen Schnittstelle anzeigen
- Einzelne Verbindungen als Text anzeigen
- Anzahl der Zyklen und Buffer anzeigen

# Sensor Array Application

Das Programm zum Empfangen und Auswerten der Daten des Sensorarrays wurde mit Microsoft Visual Studio 2010 in C# geschrieben. Es hat folgende Aufgaben/Funktionen:

- Verbinden mit unterschiedlichen COM-Ports
- Daten empfangen, auswerten
- Kontaktmatrizen und Verbindungen grafisch darstellen
- Datenstrom der seriellen Schnittstelle anzeigen
- Einzelne Verbindungen als Text anzeigen
- Anzahl der Zyklen und Buffer anzeigen

# Sensor Array Application

Das Programm zum Empfangen und Auswerten der Daten des Sensorarrays wurde mit Microsoft Visual Studio 2010 in C# geschrieben. Es hat folgende Aufgaben/Funktionen:

- Verbinden mit unterschiedlichen COM-Ports
- Daten empfangen, auswerten
- Kontaktmatrizen und Verbindungen grafisch darstellen
- Datenstrom der seriellen Schnittstelle anzeigen
- Einzelne Verbindungen als Text anzeigen
- Anzahl der Zyklen und Buffer anzeigen

# Sensor Array Application

Das Programm zum Empfangen und Auswerten der Daten des Sensorarrays wurde mit Microsoft Visual Studio 2010 in C# geschrieben. Es hat folgende Aufgaben/Funktionen:

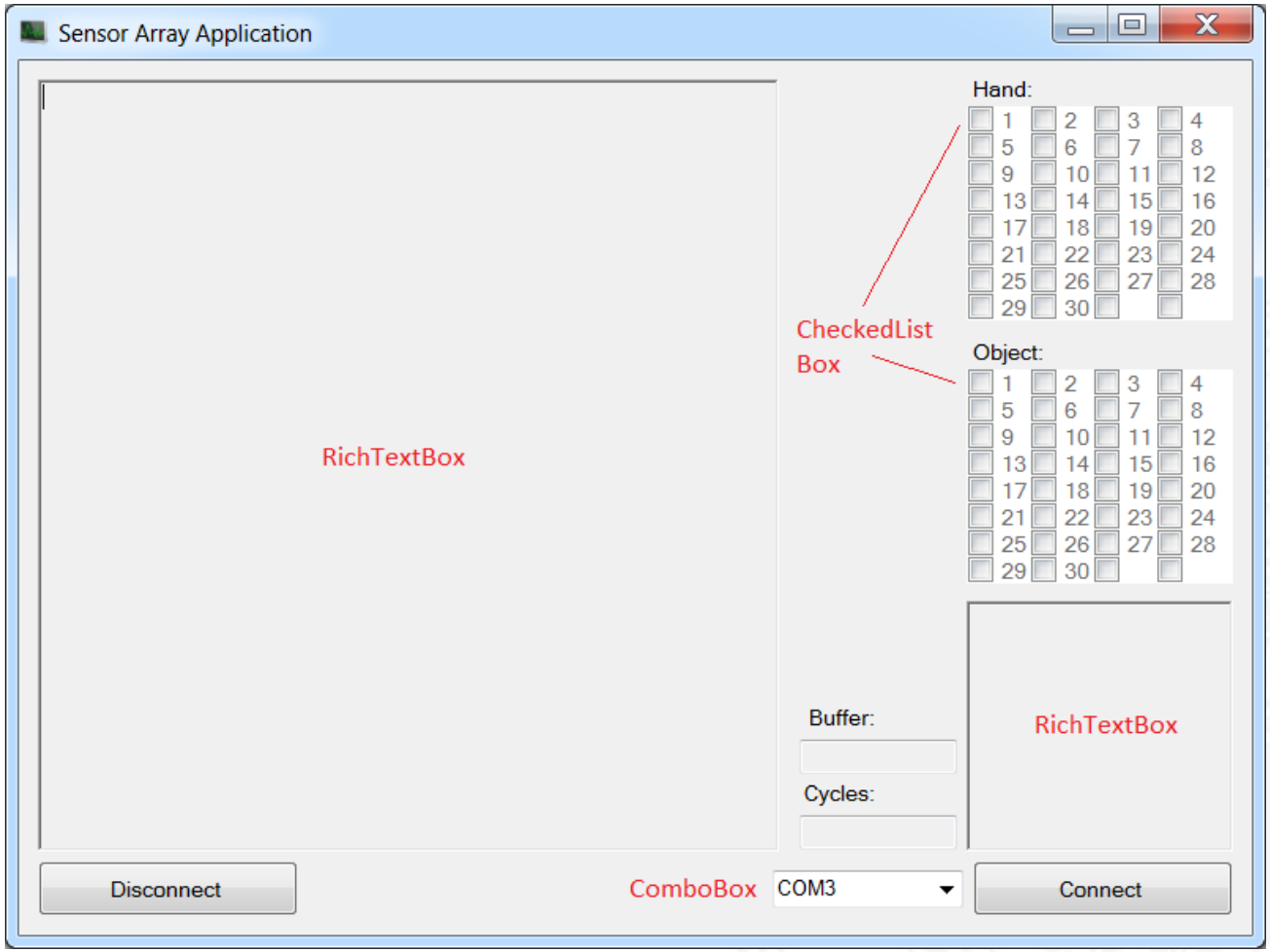
- Verbinden mit unterschiedlichen COM-Ports
- Daten empfangen, auswerten
- Kontaktmatrizen und Verbindungen grafisch darstellen
- Datenstrom der seriellen Schnittstelle anzeigen
- Einzelne Verbindungen als Text anzeigen
- Anzahl der Zyklen und Buffer anzeigen

# Sensor Array Application

Das Programm zum Empfangen und Auswerten der Daten des Sensorarrays wurde mit Microsoft Visual Studio 2010 in C# geschrieben. Es hat folgende Aufgaben/Funktionen:

- Verbinden mit unterschiedlichen COM-Ports
- Daten empfangen, auswerten
- Kontaktmatrizen und Verbindungen grafisch darstellen
- Datenstrom der seriellen Schnittstelle anzeigen
- Einzelne Verbindungen als Text anzeigen
- Anzahl der Zyklen und Buffer anzeigen

# Screenshot Sensor Array Application



# Sensor Array Application

Der wichtige Teil des Programms wird ausgeführt sobald man auf den Button Connect drückt und das richtige COM-Port in der ComboBox ausgewählt ist.

- Es wird der serielle Port initialisiert (Baudrate, Datenbits, Stopbits, etc.)
- Der serielle Port wird geöffnet und ein BackgroundWorker gestartet

# Sensor Array Application

Der wichtige Teil des Programms wird ausgeführt sobald man auf den Button Connect drückt und das richtige COM-Port in der ComboBox ausgewählt ist.

- Es wird der serielle Port initialisiert (Baudrate, Datenbits, Stopbits, etc.)
- Der serielle Port wird geöffnet und ein BackgroundWorker gestartet

# Sensor Array Application

Ablauf des BackgroundWorkers:

- Auslesen einer ganzen Zeile vom COM-Port
- Buffer löschen, falls zu groß
- Zeile je nach Präfix unterschiedlich behandeln
- Zeile im Textfeld ausgeben, ggf. Verbindungstext in anderem Textfeld anzeigen

# Sensor Array Application

Ablauf des BackgroundWorkers:

- Auslesen einer ganzen Zeile vom COM-Port
- Buffer löschen, falls zu groß
- Zeile je nach Präfix unterschiedlich behandeln
- Zeile im Textfeld ausgeben, ggf. Verbindungstext in anderem Textfeld anzeigen

# Sensor Array Application

Ablauf des BackgroundWorkers:

- Auslesen einer ganzen Zeile vom COM-Port
- Buffer löschen, falls zu groß
- Zeile je nach Präfix unterschiedlich behandeln
- Zeile im Textfeld ausgeben, ggf. Verbindungstext in anderem Textfeld anzeigen

# Sensor Array Application

Ablauf des BackgroundWorkers:

- Auslesen einer ganzen Zeile vom COM-Port
- Buffer löschen, falls zu groß
- Zeile je nach Präfix unterschiedlich behandeln
- Zeile im Textfeld ausgeben, ggf. Verbindungstext in anderem Textfeld anzeigen

# Sensor Array Application

Je nach Präfix wird vom BackgroundWorker unterschiedlich reagiert:

- Bei 'S' wird die zugehörige Checkbox des Pins von der Hand aktiviert
- Bei 'A', 'B', 'C' oder 'D' wird die jeweilige Checkbox des Pins vom Objekt aktiviert
- Bei 'N' wird der zugehörige Zyklus gesetzt und es werden alle Checkboxen resettet bis zum nächsten Zyklus
- Außerdem wird hier der Verbindungstext gelöscht, wenn die Verbindung nicht mehr vorhanden ist

# Sensor Array Application

Je nach Präfix wird vom BackgroundWorker unterschiedlich reagiert:

- Bei 'S' wird die zugehörige Checkbox des Pins von der Hand aktiviert
- Bei 'A', 'B', 'C' oder 'D' wird die jeweilige Checkbox des Pins vom Objekt aktiviert
- Bei 'N' wird der zugehörige Zyklus gesetzt und es werden alle Checkboxen resettet bis zum nächsten Zyklus
- Außerdem wird hier der Verbindungstext gelöscht, wenn die Verbindung nicht mehr vorhanden ist

# Sensor Array Application

Je nach Präfix wird vom BackgroundWorker unterschiedlich reagiert:

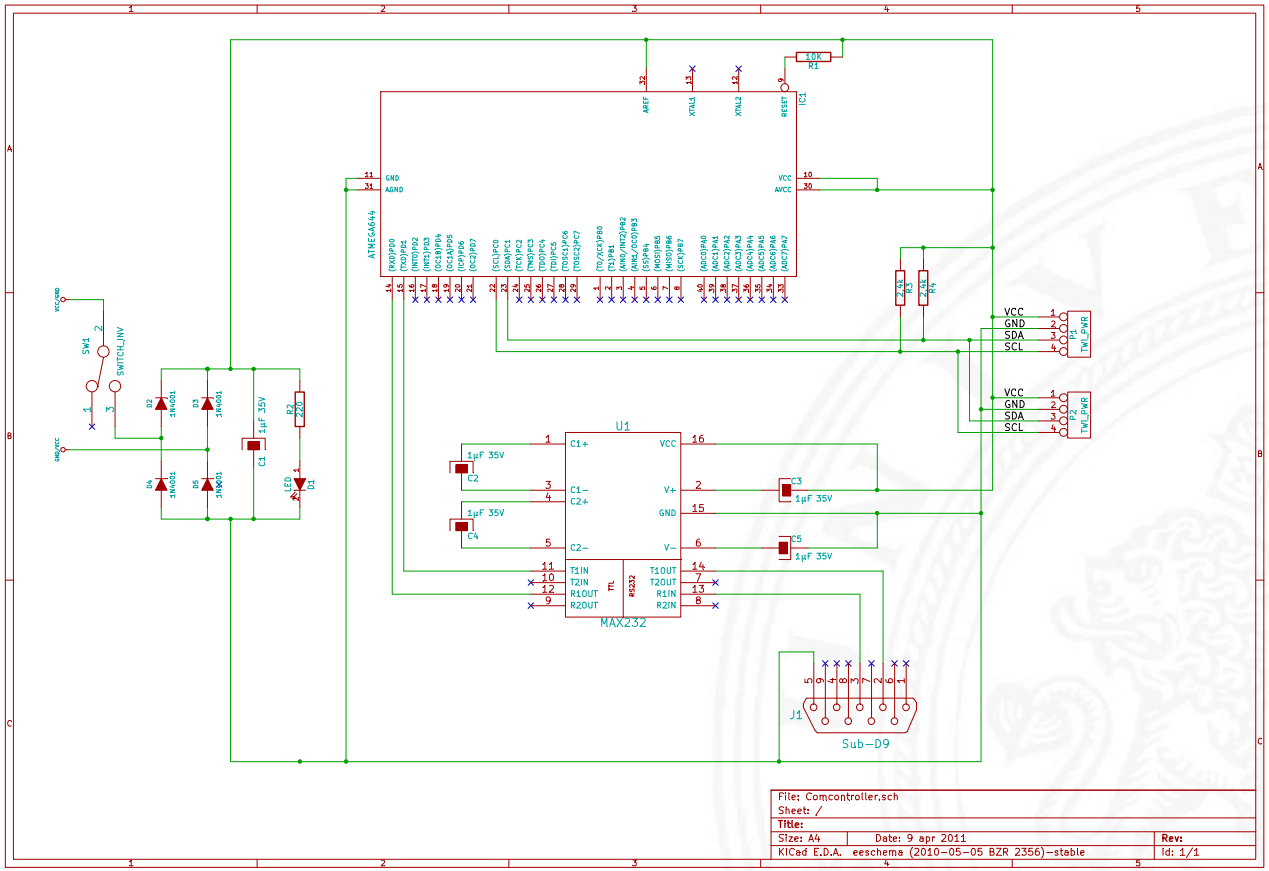
- Bei 'S' wird die zugehörige Checkbox des Pins von der Hand aktiviert
- Bei 'A', 'B', 'C' oder 'D' wird die jeweilige Checkbox des Pins vom Objekt aktiviert
- Bei 'N' wird der zugehörige Zyklus gesetzt und es werden alle Checkboxen resettet bis zum nächsten Zyklus
- Außerdem wird hier der Verbindungstext gelöscht, wenn die Verbindung nicht mehr vorhanden ist

# Sensor Array Application

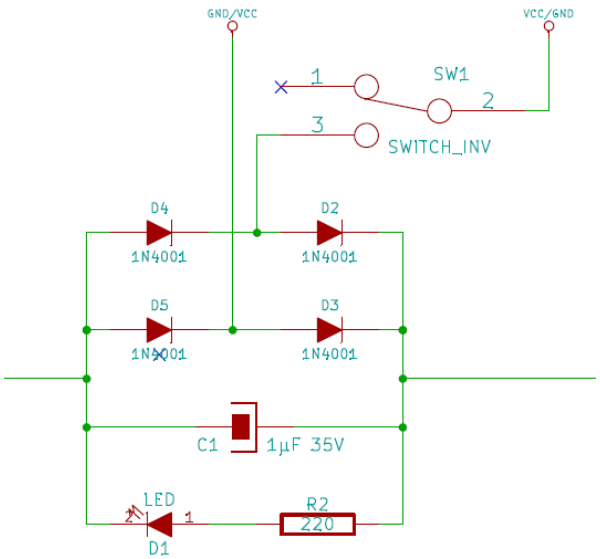
Je nach Präfix wird vom BackgroundWorker unterschiedlich reagiert:

- Bei 'S' wird die zugehörige Checkbox des Pins von der Hand aktiviert
- Bei 'A', 'B', 'C' oder 'D' wird die jeweilige Checkbox des Pins vom Objekt aktiviert
- Bei 'N' wird der zugehörige Zyklus gesetzt und es werden alle Checkboxen resettet bis zum nächsten Zyklus
- Außerdem wird hier der Verbindungstext gelöscht, wenn die Verbindung nicht mehr vorhanden ist

# ComController-Schaltung



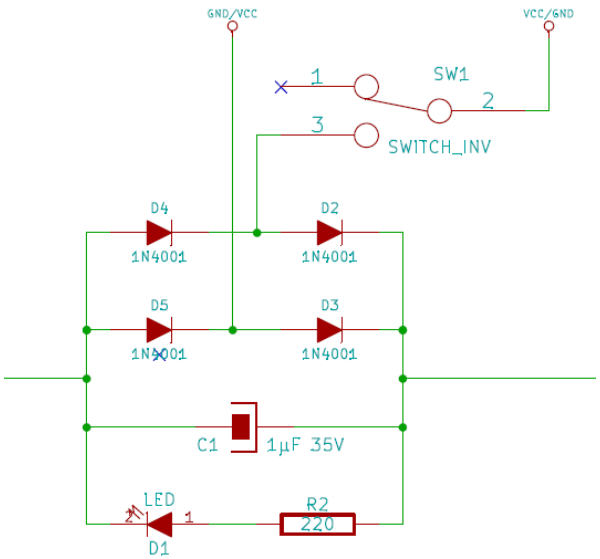
# Der Eingangsbereich




Der Eingangsbereich der ComController-Schaltung besteht aus mehreren Komponenten:

- Ein Schalter zum ein- und ausschalten des Systems.
- Der Verpolschutz besteht aus einer Brückenschaltung mit vier 1N4001 Siliziumdioden.
- Zur Spannungsstabilisierung bzw. Glättung dient der $1\mu$F Kondensator.
- Eine Betriebsdiode zeigt ob das System ein- oder ausgeschaltet ist.

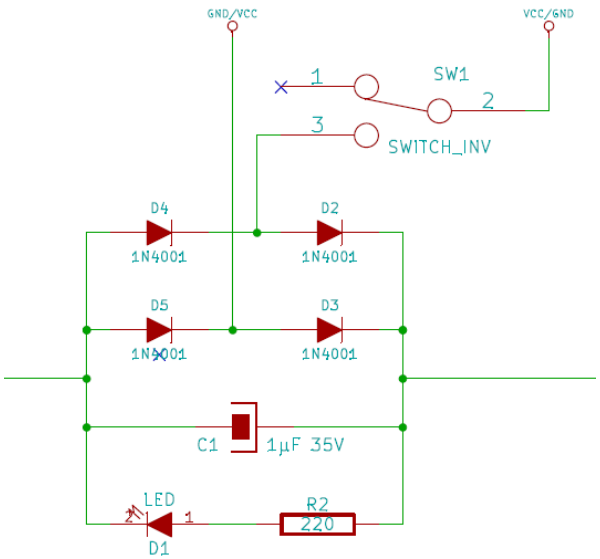
# Der Eingangsbereich




Der Eingangsbereich der ComController-Schaltung besteht aus mehreren Komponenten:

- Ein Schalter zum ein- und ausschalten des Systems.
- Der Verpolschutz besteht aus einer Brückenschaltung mit vier 1N4001 Siliziumdioden.
- Zur Spannungsstabilisierung bzw. Glättung dient der $1\mu$F Kondensator.
- Eine Betriebsdiode zeigt ob das System ein- oder ausgeschaltet ist.

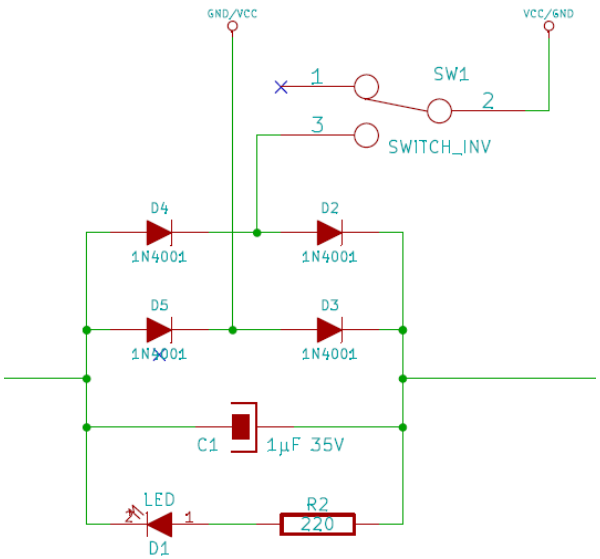
# Der Eingangsbereich




Der Eingangsbereich der ComController-Schaltung besteht aus mehreren Komponenten:

- Ein Schalter zum ein- und ausschalten des Systems.
- Der Verpolschutz besteht aus einer Brückenschaltung mit vier 1N4001 Siliziumdioden.
- Zur Spannungsstabilisierung bzw. Glättung dient der 1$\mu$F Kondensator.
- Eine Betriebsdiode zeigt ob das System ein- oder ausgeschaltet ist.

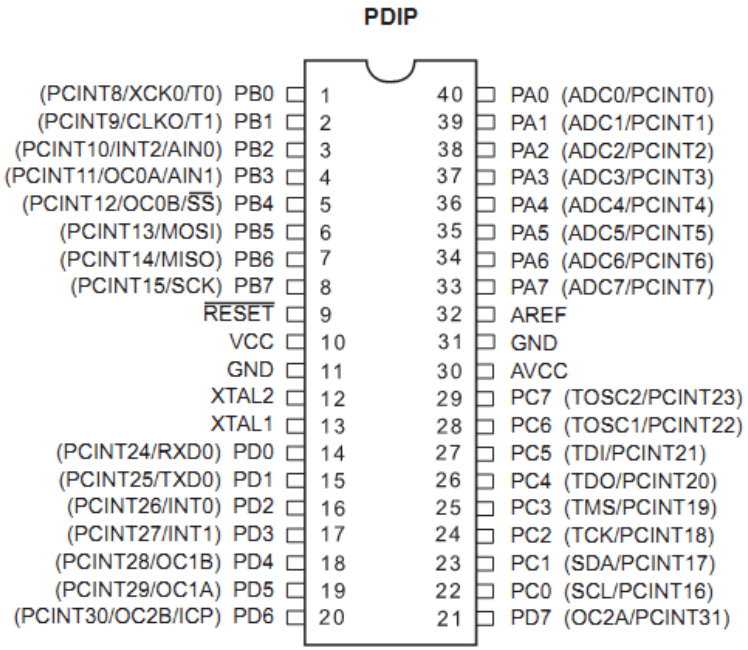
# Der Eingangsbereich




Der Eingangsbereich der ComController-Schaltung besteht aus mehreren Komponenten:

- Ein Schalter zum ein- und ausschalten des Systems.
- Der Verpolschutz besteht aus einer Brückenschaltung mit vier 1N4001 Siliziumdioden.
- Zur Spannungsstabilisierung bzw. Glättung dient der 1$\mu$F Kondensator.
- Eine Betriebsdiode zeigt ob das System ein- oder ausgeschaltet ist.

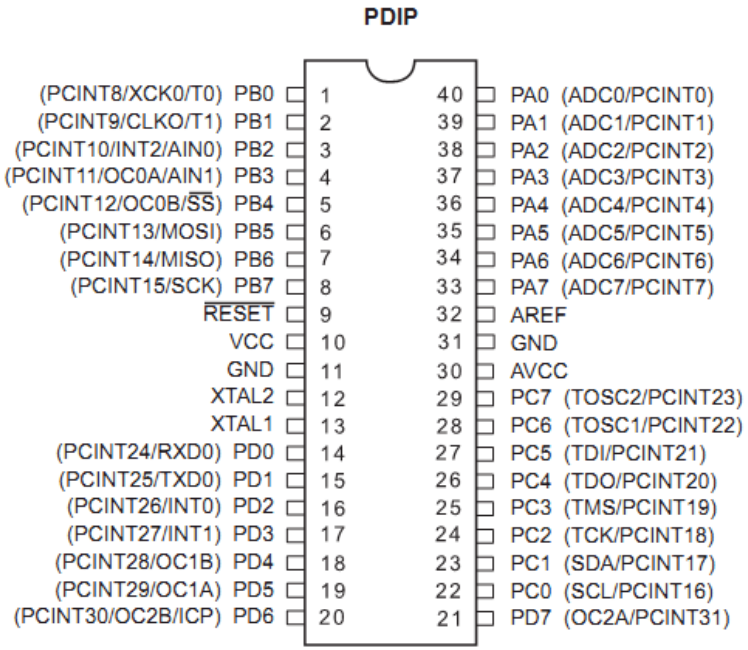
# Atmel® ATmega644-20 PU




Die wichtigsten Eigenschaften des eingesetzten Mikrocontroller:

- 40Pin DIL-Gehäuse.
- 32 Ein-/Ausgänge in 4 Ports.
- Interner Quarz mit 8Mhz.
- Integrierte USART-Schnittstelle.
- Integrierter TWI-Bus

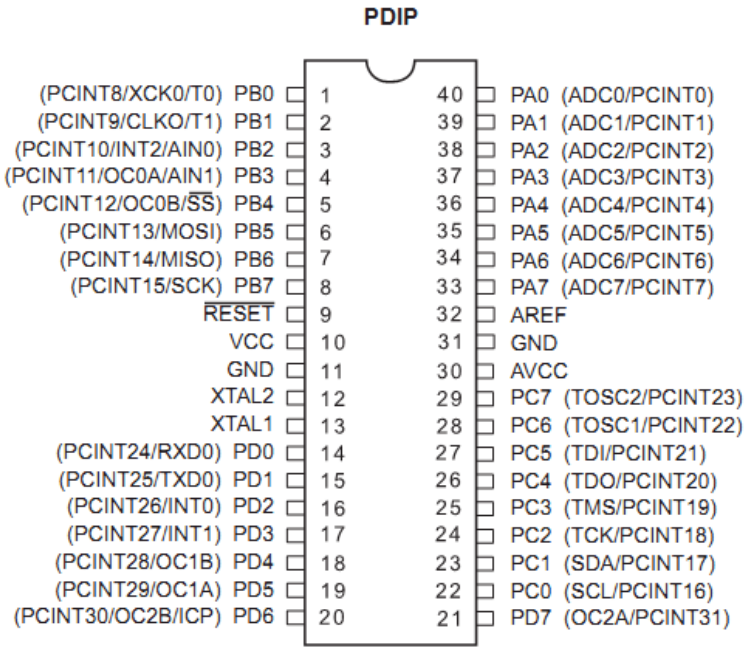
# Atmel®ATmega644-20 PU




Die wichtigsten Eigenschaften des eingesetzten Mikrocontroller:

- 40Pin DIL-Gehäuse.
- 32 Ein-/Ausgänge in 4 Ports.
- Interner Quarz mit 8Mhz.
- Integrierte USART-Schnittstelle.
- Integrierter TWI-Bus

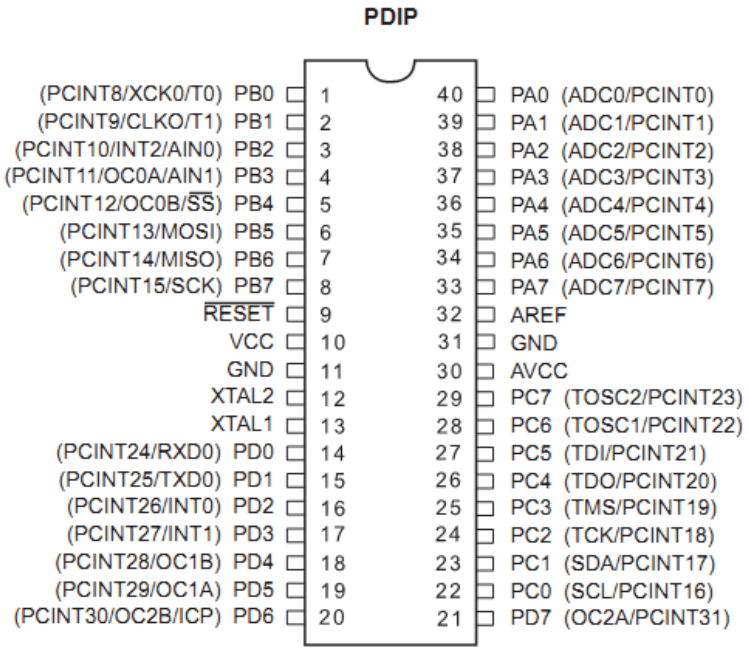
# Atmel®ATmega644-20 PU




Die wichtigsten Eigenschaften des eingesetzten Mikrocontroller:

- 40Pin DIL-Gehäuse.
- 32 Ein-/Ausgänge in 4 Ports.
- Interner Quarz mit 8Mhz.
- Integrierte USART-Schnittstelle.
- Integrierter TWI-Bus

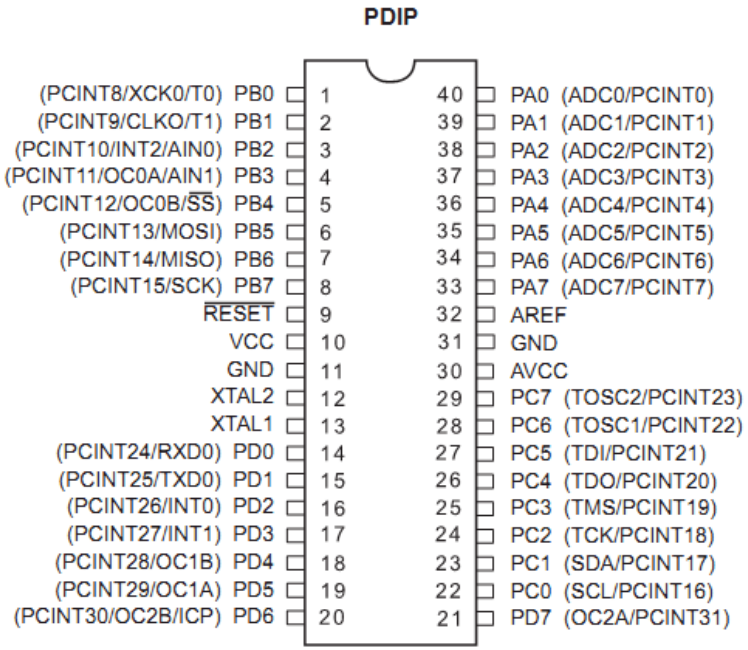
# Atmel®ATmega644-20 PU




Die wichtigsten Eigenschaften des eingesetzten Mikrocontroller:

- 40Pin DIL-Gehäuse.
- 32 Ein-/Ausgänge in 4 Ports.
- Interner Quarz mit 8Mhz.
- Integrierte USART-Schnittstelle.
- Integrierter TWI-Bus

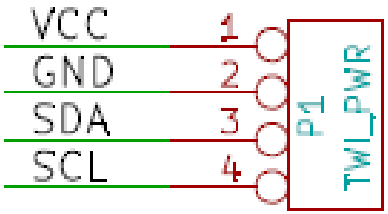
# Atmel®ATmega644-20 PU




Die wichtigsten Eigenschaften des eingesetzten Mikrocontroller:

- 40Pin DIL-Gehäuse.
- 32 Ein-/Ausgänge in 4 Ports.
- Interner Quarz mit 8Mhz.
- Integrierte USART-Schnittstelle.
- Integrierter TWI-Bus

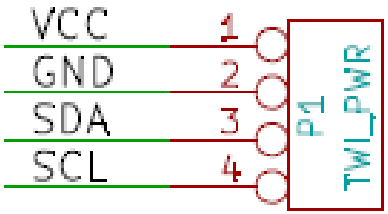
# TWI/PWR




Der Twi-Bus stellt die Kommunikation zwischen den einzelnen Modulen.

- 100kHz bis 400kHz
- High-Level Spannung auf SDA und SCL
- daher 2,4kΩ Widerstände
- Stromversorgung hinzugefügt
- Verpolschutz durch Unanfälligkeit

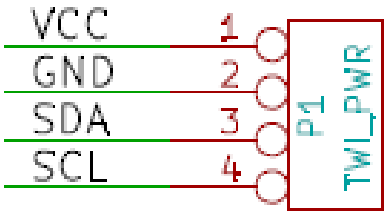
# TWI/PWR




Der Twi-Bus stellt die Kommunikation zwischen den einzelnen Modulen.

- 100kHz bis 400kHz
- High-Level Spannung auf SDA und SCL
- daher 2,4kΩ Widerstände
- Stromversorgung hinzugefügt
- Verpolschutz durch Unanfälligkeit

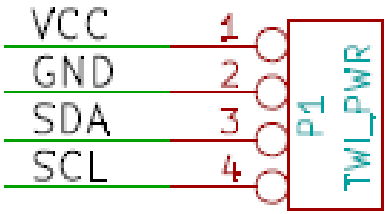
# TWI/PWR




Der Twi-Bus stellt die Kommunikation zwischen den einzelnen Modulen.

- 100kHz bis 400kHz
- High-Level Spannung auf SDA und SCL
- daher 2,4kΩ Widerstände
- Stromversorgung hinzugefügt
- Verpolschutz durch Unanfälligkeit

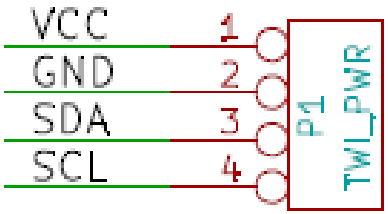
# TWI/PWR




Der Twi-Bus stellt die Kommunikation zwischen den einzelnen Modulen.

- 100kHz bis 400kHz
- High-Level Spannung auf SDA und SCL
- daher 2,4kΩ Widerstände
- Stromversorgung hinzugefügt
- Verpolschutz durch Unanfälligkeit

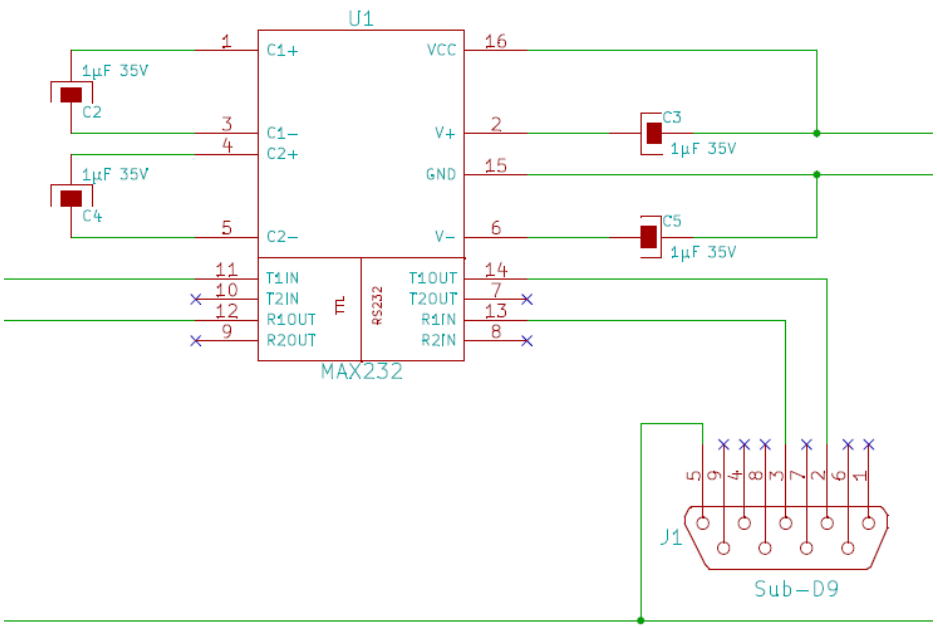
# TWI/PWR




Der Twi-Bus stellt die Kommunikation zwischen den einzelnen Modulen.

- 100kHz bis 400kHz
- High-Level Spannung auf SDA und SCL
- daher 2,4kΩ Widerstände
- Stromversorgung hinzugefügt
- Verpolschutz durch Unanfälligkeit

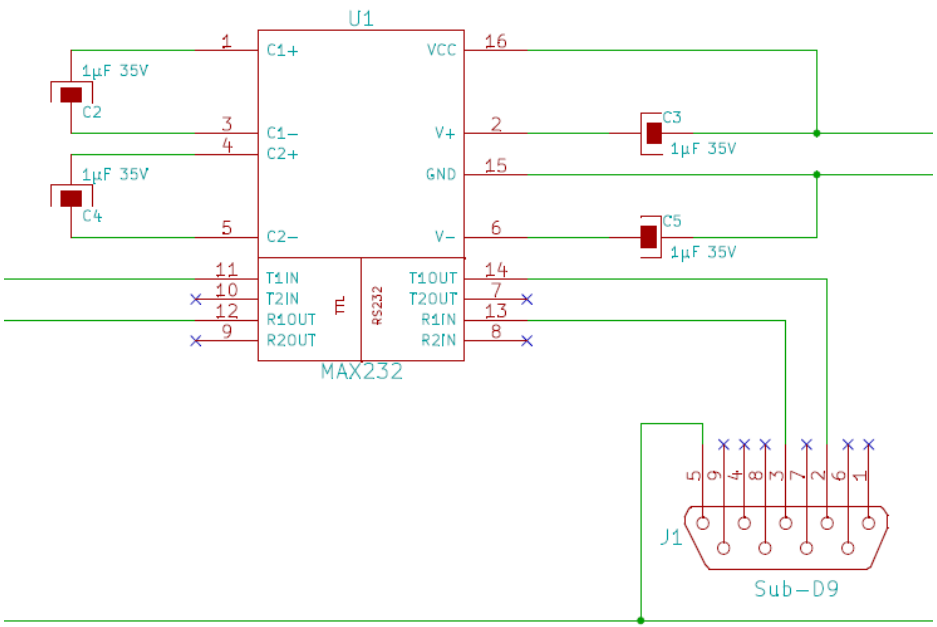
# RS-232




Die RS-232 Schnittstelle bildet die Kommunikation zum Rechner.

- MAX232 von Texas Instruments
- 16Pin DIL-Gehäuse
- Umsetzung von 5V auf 12V
- USART vorgegeben
- Zwei Leitungen

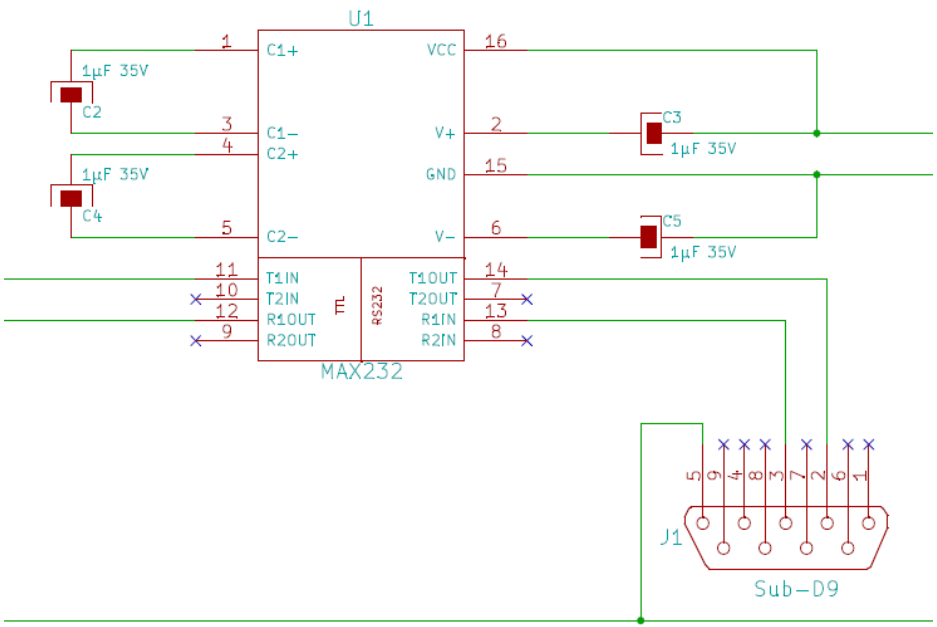
# RS-232




Die RS-232 Schnittstelle bildet die Kommunikation zum Rechner.

- MAX232 von Texas Instruments
- 16Pin DIL-Gehäuse
- Umsetzung von 5V auf 12V
- USART vorgegeben
- Zwei Leitungen

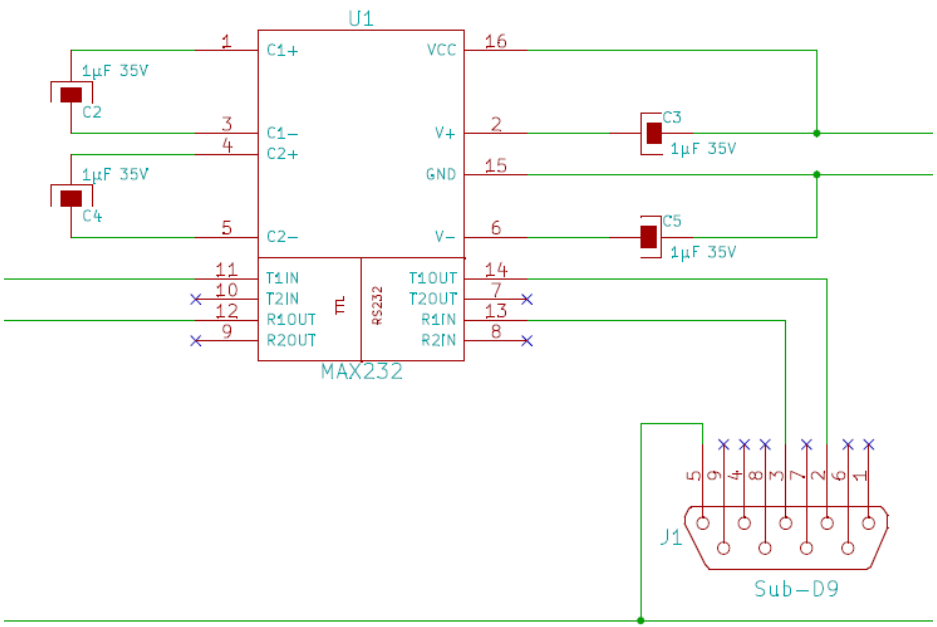
# RS-232




Die RS-232 Schnittstelle bildet die Kommunikation zum Rechner.

- MAX232 von Texas Instruments
- 16Pin DIL-Gehäuse
- Umsetzung von 5V auf 12V
- USART vorgegeben
- Zwei Leitungen

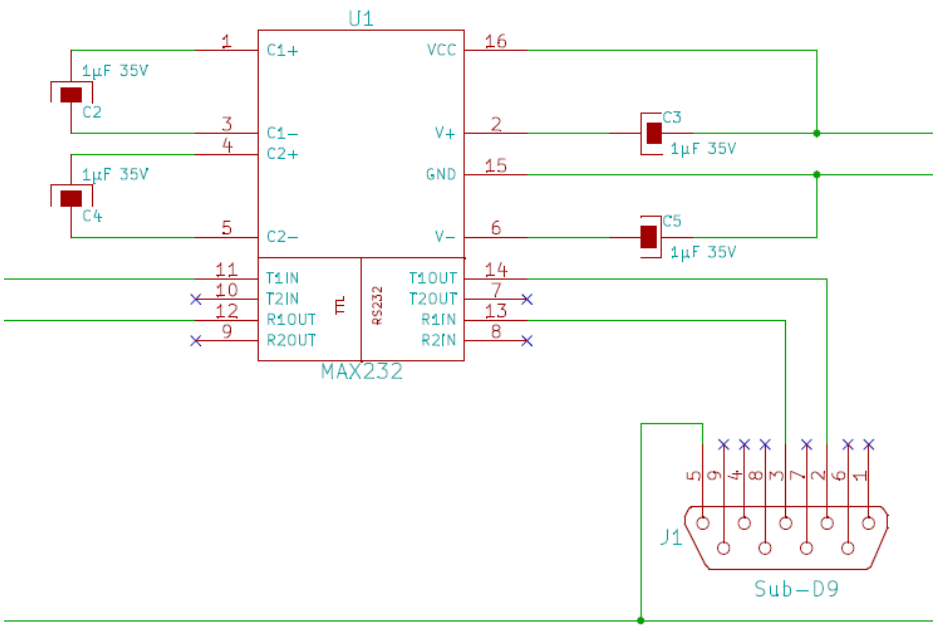
# RS-232




Die RS-232 Schnittstelle bildet die Kommunikation zum Rechner.

- MAX232 von Texas Instruments
- 16Pin DIL-Gehäuse
- Umsetzung von 5V auf 12V
- USART vorgegeben
- Zwei Leitungen

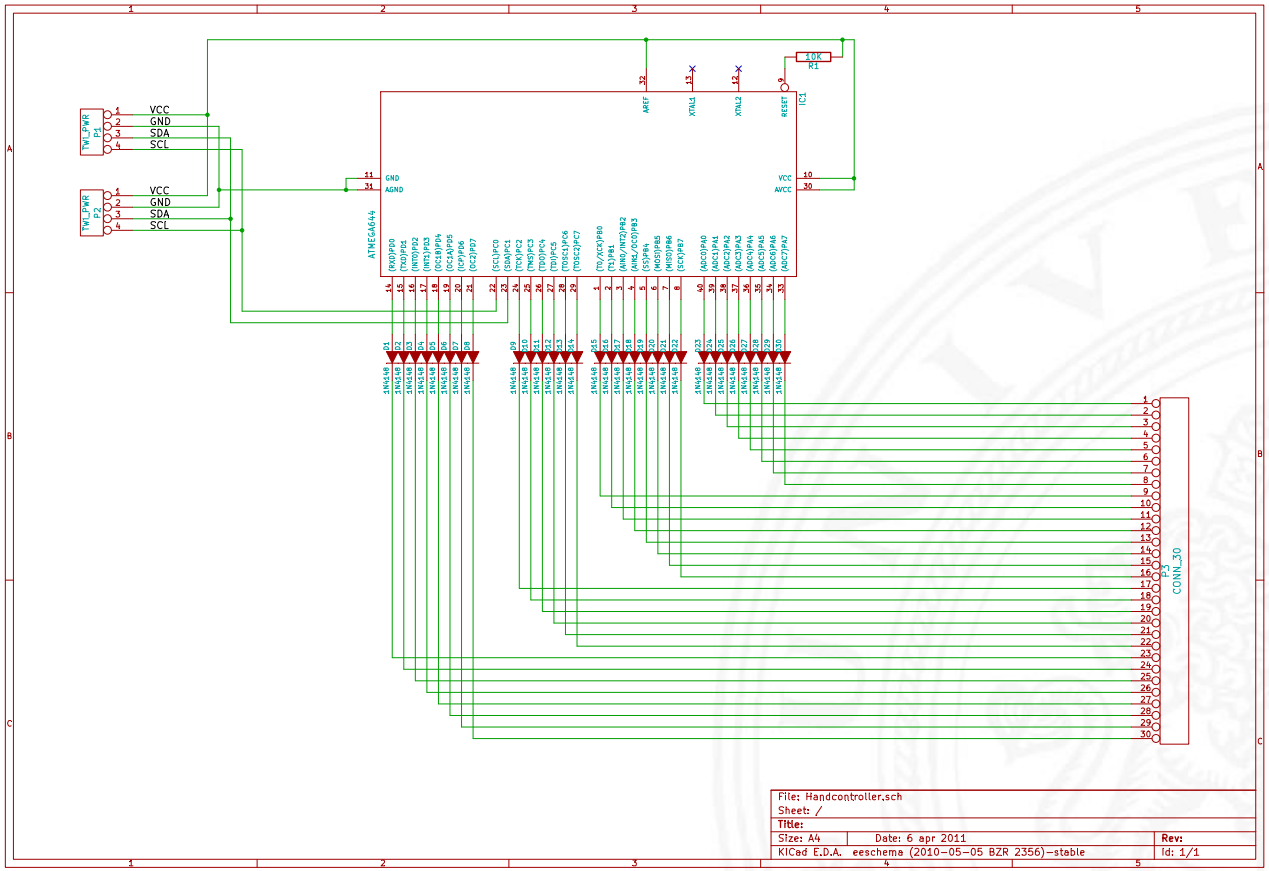
# RS-232



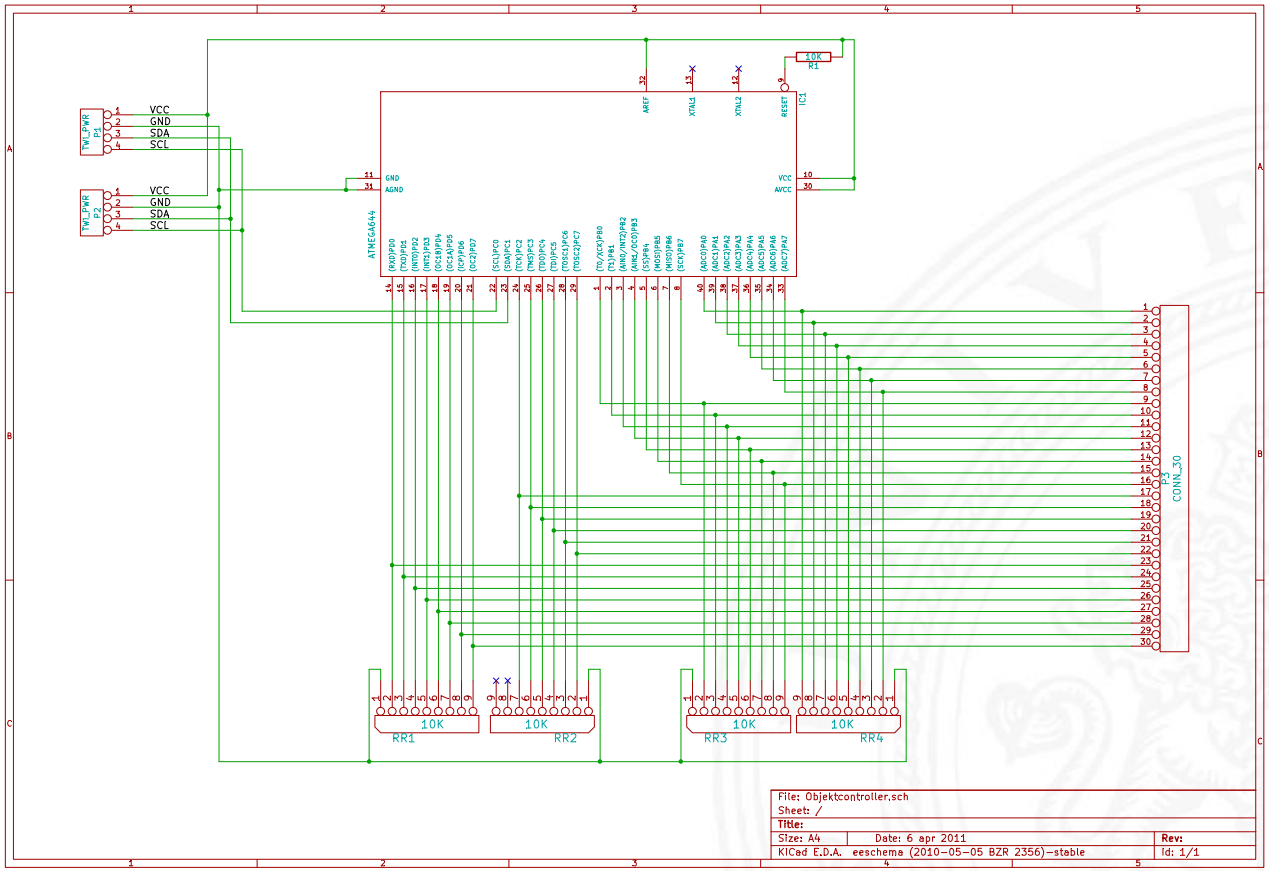

Die RS-232 Schnittstelle bildet die Kommunikation zum Rechner.

- MAX232 von Texas Instruments
- 16Pin DIL-Gehäuse
- Umsetzung von 5V auf 12V
- USART vorgegeben
- Zwei Leitungen

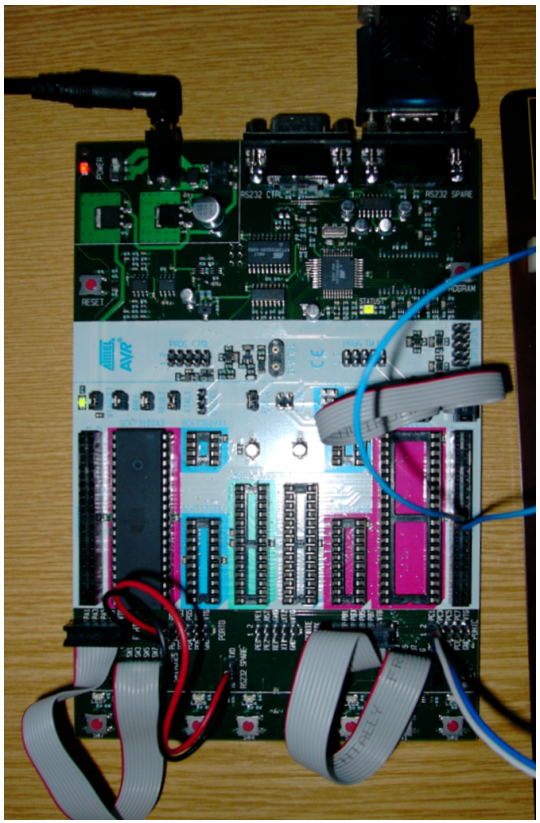
# HandController-Schaltung



# ObjektController-Schaltung



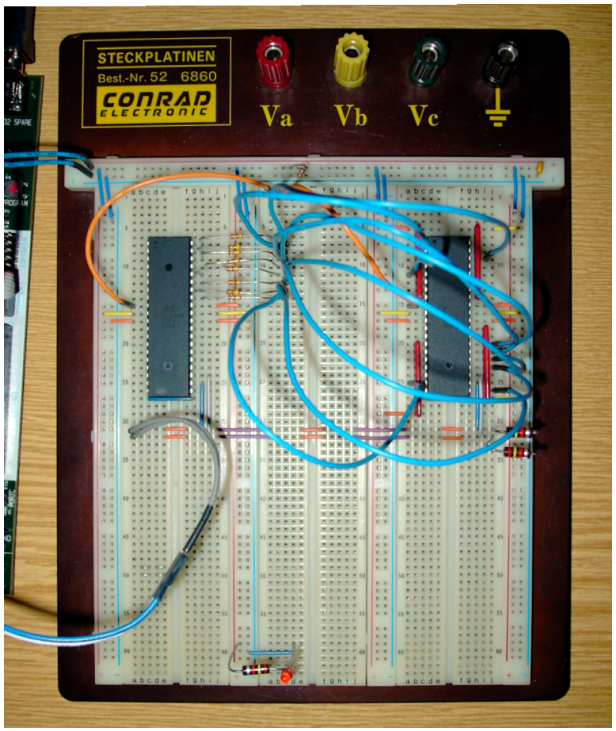
# AVR®STK500



Das STK500 diente als Einstieg in die Welt der Mikrocontroller.

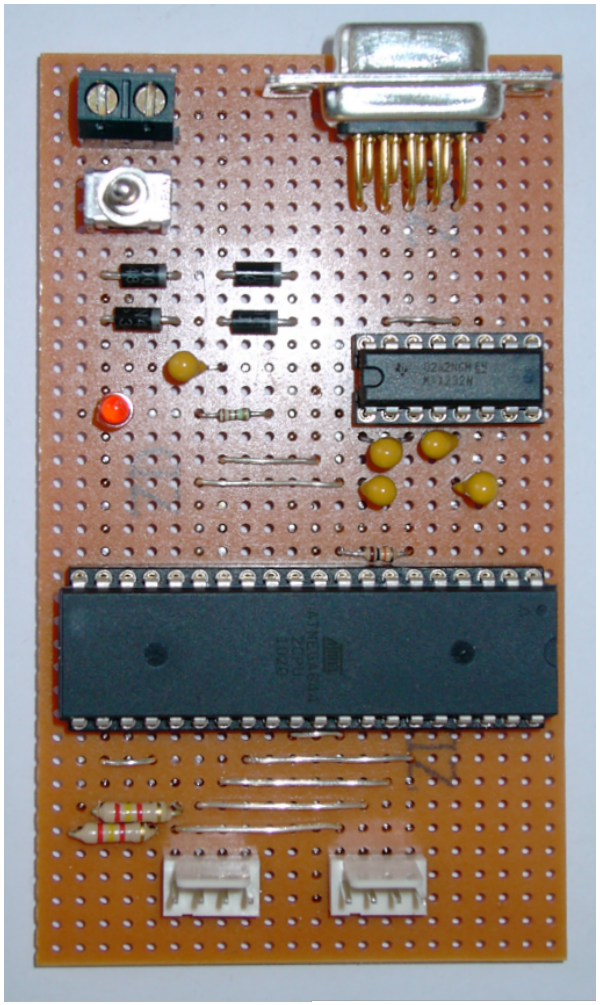
- Programmieren der ATmega644-20 PU.
- Ausprobieren der Programmierung.
- RS-232 Schnittstelle vorhanden.

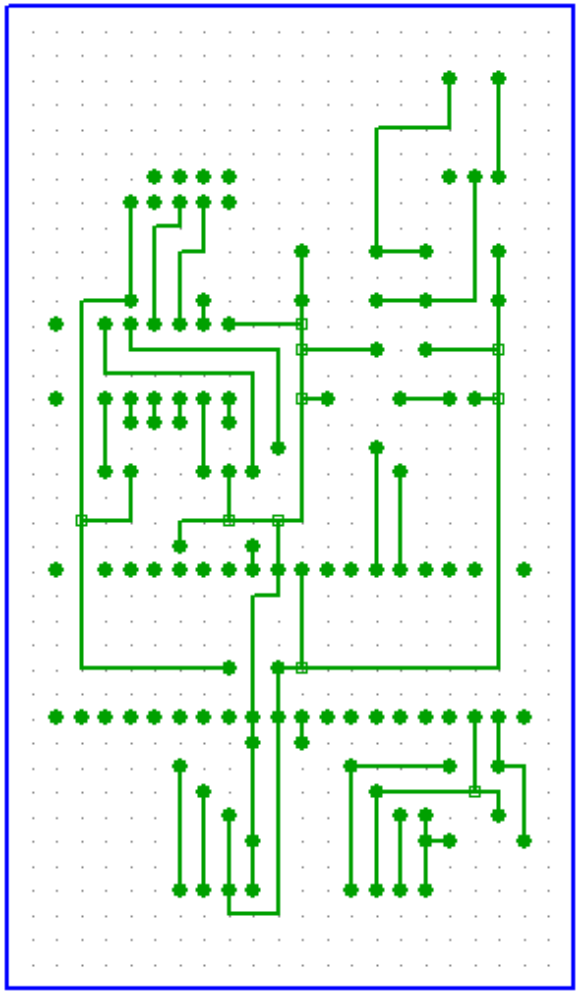
# Erweiterungssteckplatine



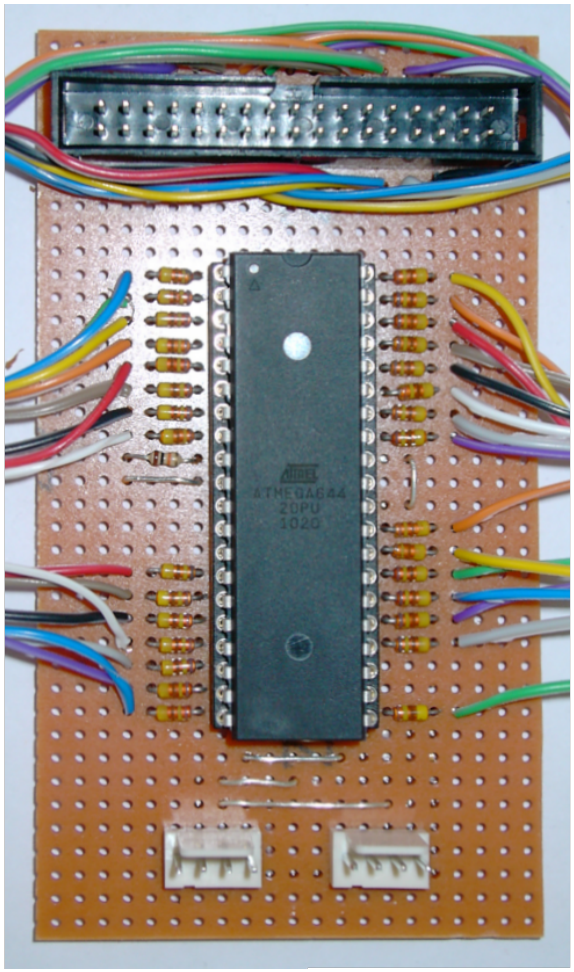

Ereweiterung des STK500.
Simulation der Kontaktmatrix über Kabel.
Links: HandController mit 8 belegten Ausgängen, an Port A.
Rechts: ObjektController mit 8 belegten unterschiedlichen Eingängen.

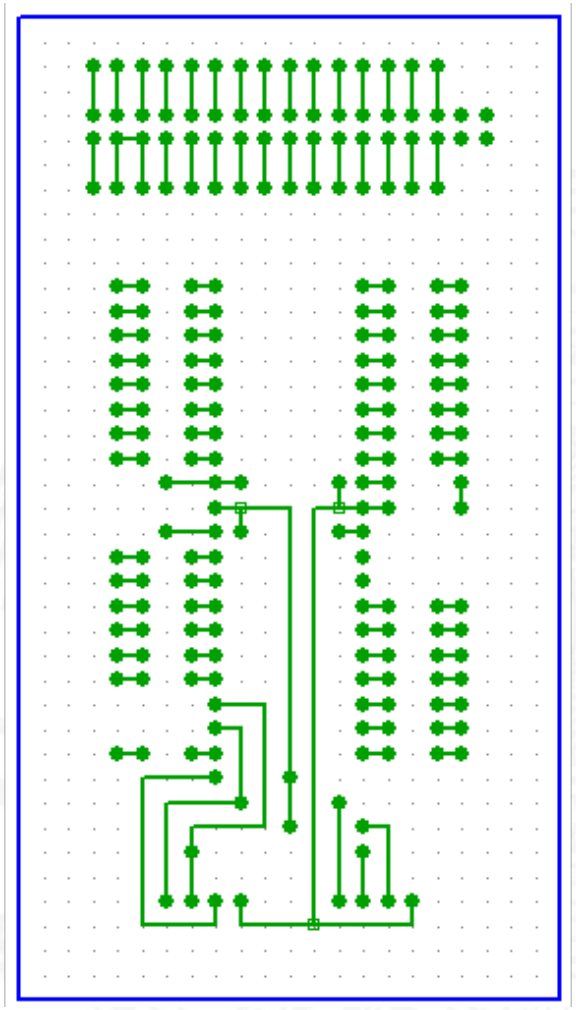
# ComController



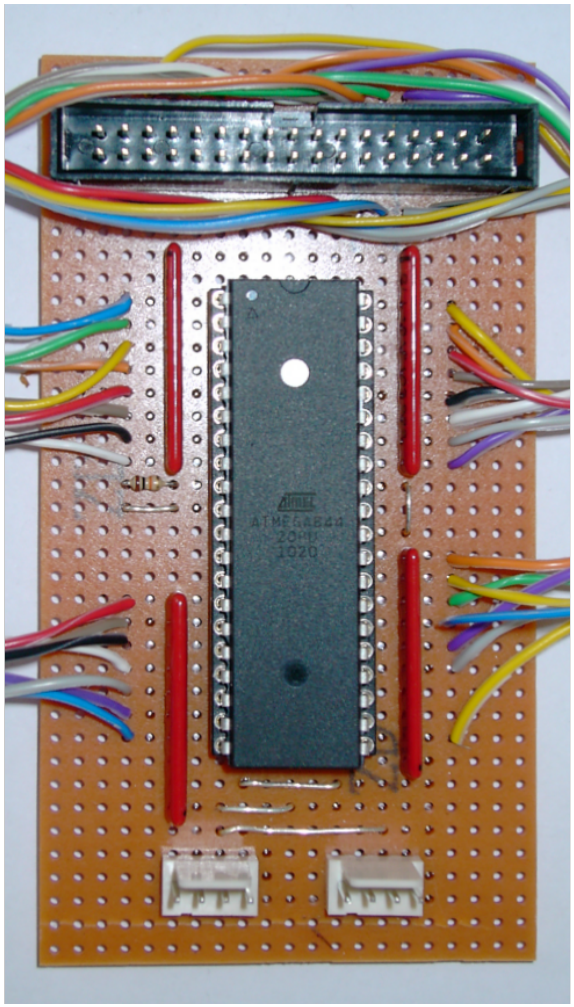


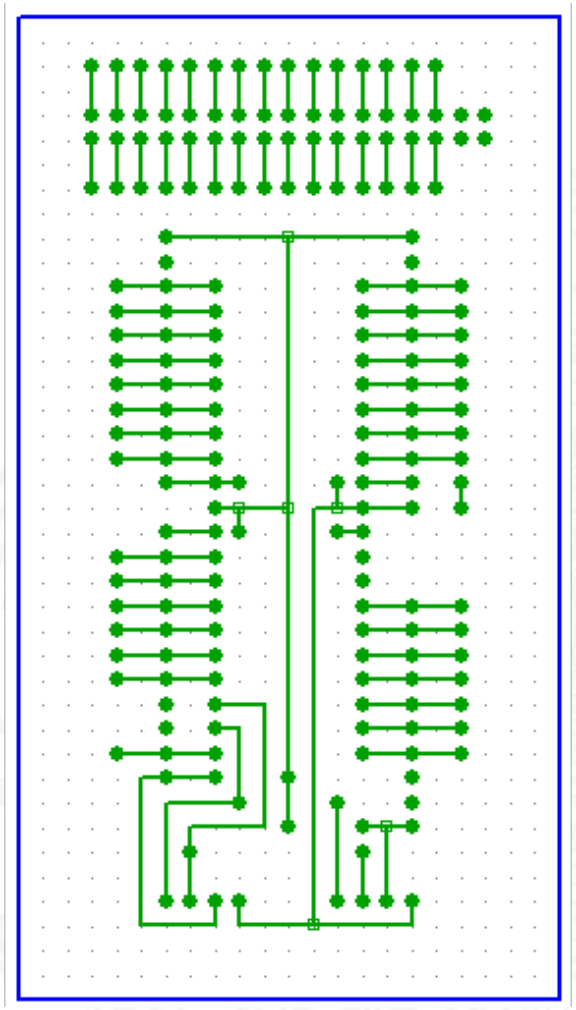
# HandController



# ObjektController





# Kontaktmatrix

Die Kontaktmatrix eines Controllers ist die Schnittstelle zu einer weiteren Kontaktmatrix.
Dabei erfüllt sie folgende Eigenschaften:

- Besteht aus Kontakten.
- Matrixförmige Anordnung.
- Dient der Positionsbestimmung.
- Mehre Umsetzungsmöglichkeiten.

# Kontaktmatrix

Die Kontaktmatrix eines Controllers ist die Schnittstelle zu einer weiteren Kontaktmatrix.
Dabei erfüllt sie folgende Eigenschaften:

- Besteht aus Kontakten.
- Matrixförmige Anordnung.
- Dient der Positionsbestimmung.
- Mehre Umsetzungsmöglichkeiten.

# Kontaktmatrix

Die Kontaktmatrix eines Controllers ist die Schnittstelle zu einer weiteren Kontaktmatrix.
Dabei erfüllt sie folgende Eigenschaften:

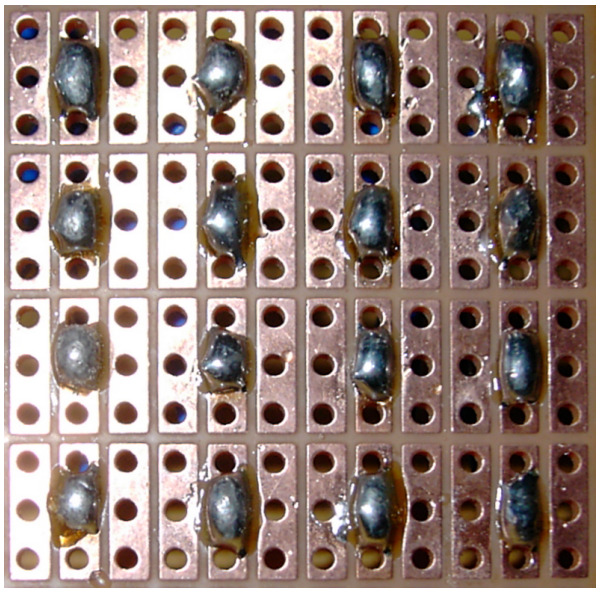
- Besteht aus Kontakten.
- Matrixförmige Anordnung.
- Dient der Positionsbestimmung.
- Mehre Umsetzungsmöglichkeiten.

# Kontaktmatrix

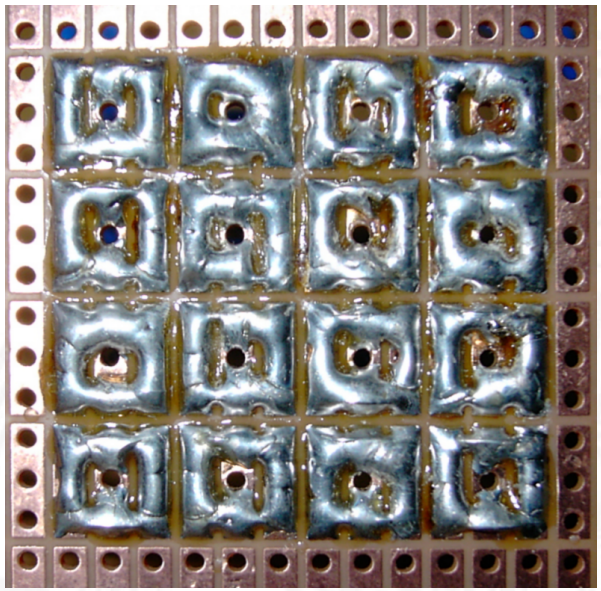
Die Kontaktmatrix eines Controllers ist die Schnittstelle zu einer weiteren Kontaktmatrix.
Dabei erfüllt sie folgende Eigenschaften:

- Besteht aus Kontakten.
- Matrixförmige Anordnung.
- Dient der Positionsbestimmung.
- Mehre Umsetzungsmöglichkeiten.

# Einfache Kontaktmatrix

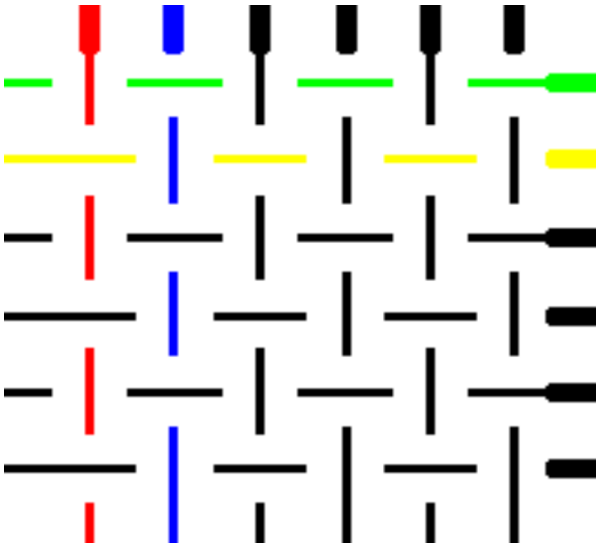


SensorMatrix für den HandController.

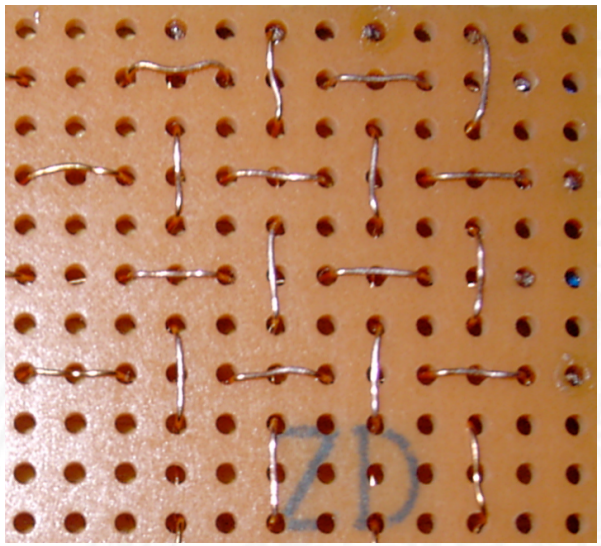


SensorMatrix für den ObjektController.

# Gewebematrix



Schema der Gewebestruktur.



Umsetzung mit Kupferlackdraht auf RM 3,5mm.

# Der Handschuh

Die Sensorfläche auf eine Hand abzubilden ist nun der nächste Schritt. Dies ist am Besten mit einem Handschuh möglich. Der Prototyp eines solchen Handschuhs ist im Rahmen der Bachelorarbeit konstruiert worden

- Einfacher Baumwollhandschuh.
- Sechs Kontakte am Zeigefinger.
- Bestehend aus Heftklammern.
- Mit Kupferlackdraht verlötet und verklebt.
- Anschluss über 34er Flachbandkabel.

# Der Handschuh

Die Sensorfläche auf eine Hand abzubilden ist nun der nächste Schritt. Dies ist am Besten mit einem Handschuh möglich. Der Prototyp eines solchen Handschuhs ist im Rahmen der Bachelorarbeit konstruiert worden

- Einfacher Baumwollhandschuh.
- Sechs Kontakte am Zeigefinger.
- Bestehend aus Heftklammern.
- Mit Kupferlackdraht verlötet und verklebt.
- Anschluss über 34er Flachbandkabel.

# Der Handschuh

Die Sensorfläche auf eine Hand abzubilden ist nun der nächste Schritt. Dies ist am Besten mit einem Handschuh möglich. Der Prototyp eines solchen Handschuhs ist im Rahmen der Bachelorarbeit konstruiert worden

- Einfacher Baumwollhandschuh.
- Sechs Kontakte am Zeigefinger.
- Bestehend aus Heftklammern.
- Mit Kupferlackdraht verlötet und verklebt.
- Anschluss über 34er Flachbandkabel.

# Der Handschuh

Die Sensorfläche auf eine Hand abzubilden ist nun der nächste Schritt. Dies ist am Besten mit einem Handschuh möglich. Der Prototyp eines solchen Handschuhs ist im Rahmen der Bachelorarbeit konstruiert worden

- Einfacher Baumwollhandschuh.
- Sechs Kontakte am Zeigefinger.
- Bestehend aus Heftklammern.
- Mit Kupferlackdraht verlötet und verklebt.
- Anschluss über 34er Flachbandkabel.

# Der Handschuh

Die Sensorfläche auf eine Hand abzubilden ist nun der nächste Schritt. Dies ist am Besten mit einem Handschuh möglich. Der Prototyp eines solchen Handschuhs ist im Rahmen der Bachelorarbeit konstruiert worden

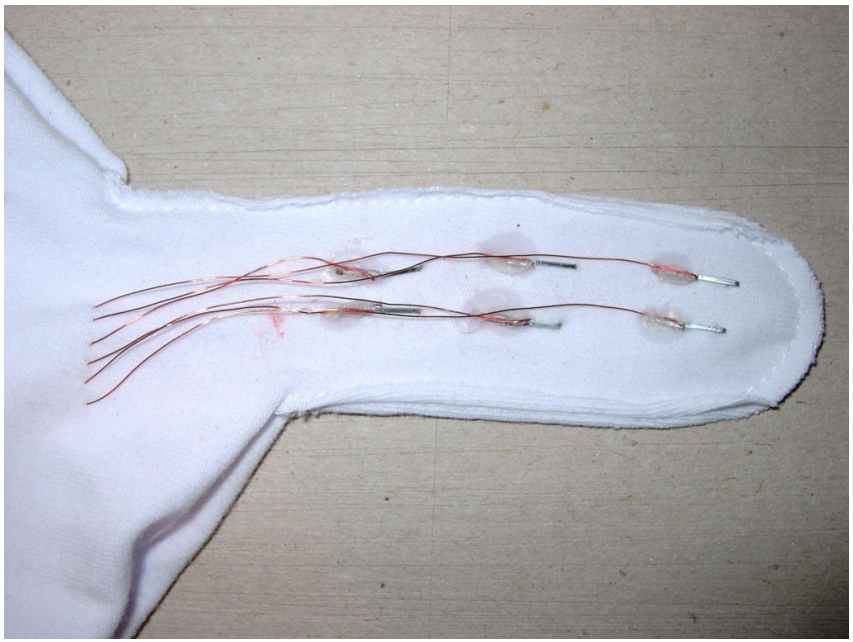
- Einfacher Baumwollhandschuh.
- Sechs Kontakte am Zeigefinger.
- Bestehend aus Heftklammern.
- Mit Kupferlackdraht verlötet und verklebt.
- Anschluss über 34er Flachbandkabel.

# Der Handschuh

Die Sensorfläche auf eine Hand abzubilden ist nun der nächste Schritt. Dies ist am Besten mit einem Handschuh möglich. Der Prototyp eines solchen Handschuhs ist im Rahmen der Bachelorarbeit konstruiert worden

- Einfacher Baumwollhandschuh.
- Sechs Kontakte am Zeigefinger.
- Bestehend aus Heftklammern.
- Mit Kupferlackdraht verlötet und verklebt.
- Anschluss über 34er Flachbandkabel.

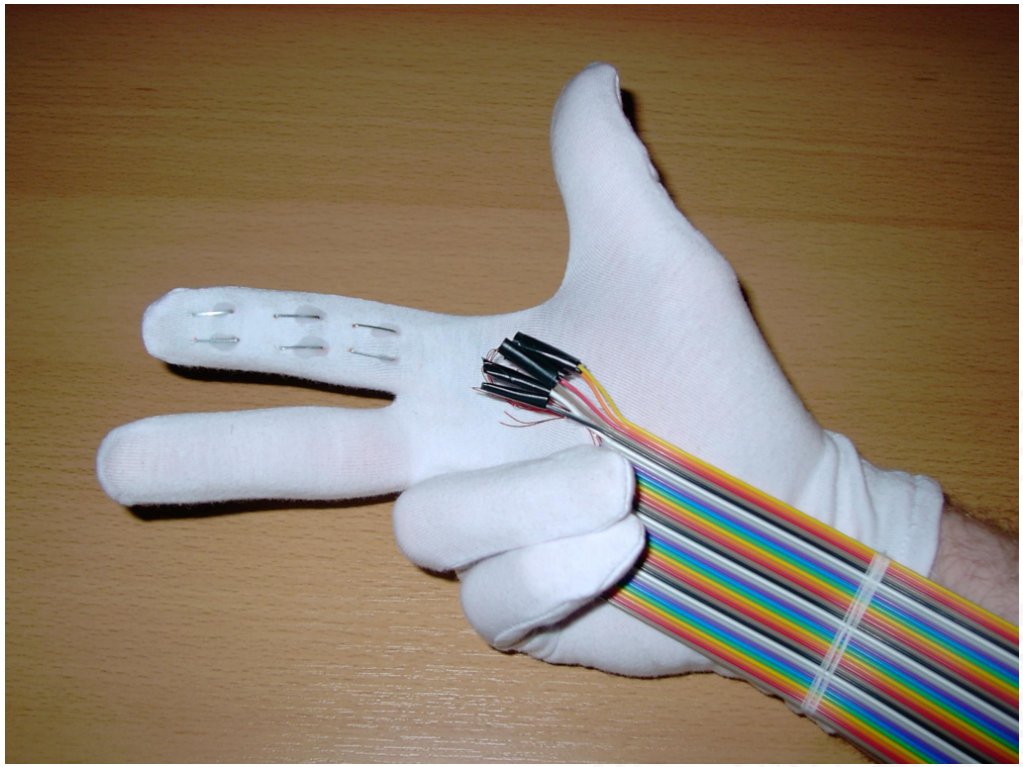
# Der Handschuh (Aufbau)



Kontakte de Handschuh an der Innenseite des Zeigefingers.

Zusätzlichen Schutz vor ungewollter Kontaktierung und vor Abriss bietet das Klebeband.

# Der Handschuh (Fertig)



Der fertige Handschuh als SensorMatrix des HandControllers.

# Das Gewebematrixtuch

Die Oberfläche eines Objektes mit Kontakten auszustatten ist notwendig. Denkbar ist hier eine Art Tuch mit das Objekt eingehüllt wird. Ein solches Tuch ist im Rahmen der Bachelorarbeit hergestellt worden.

- Einfaches Fensterleder.
- Kontaktfläche von 16cm × 16cm.
- Mit 30 eingewebten Kupferlackdrähten.
- Entstehen 225 Kontaktkoordinaten.
- Anschluss über 34er Flachbandkabel.

# Das Gewebematrixtuch

Die Oberfläche eines Objektes mit Kontakten auszustatten ist notwendig. Denkbar ist hier eine Art Tuch mit das Objekt eingehüllt wird. Ein solches Tuch ist im Rahmen der Bachelorarbeit hergestellt worden.

- Einfaches Fensterleder.
- Kontaktfläche von 16cm $\times$ 16cm.
- Mit 30 eingewebten Kupferlackdrähten.
- Entstehen 225 Kontaktkoordinaten.
- Anschluss über 34er Flachbandkabel.

# Das Gewebematrixtuch

Die Oberfläche eines Objektes mit Kontakten auszustatten ist notwendig. Denkbar ist hier eine Art Tuch mit das Objekt eingehüllt wird. Ein solches Tuch ist im Rahmen der Bachelorarbeit hergestellt worden.

- Einfaches Fensterleder.
- Kontaktfläche von 16cm $\times$ 16cm.
- Mit 30 eingewebten Kupferlackdrähten.
- Entstehen 225 Kontaktkoordinaten.
- Anschluss über 34er Flachbandkabel.

# Das Gewebematrixtuch

Die Oberfläche eines Objektes mit Kontakten auszustatten ist notwendig. Denkbar ist hier eine Art Tuch mit das Objekt eingehüllt wird. Ein solches Tuch ist im Rahmen der Bachelorarbeit hergestellt worden.

- Einfaches Fensterleder.
- Kontaktfläche von 16cm $\times$ 16cm.
- Mit 30 eingewebten Kupferlackdrähten.
- Entstehen 225 Kontaktkoordinaten.
- Anschluss über 34er Flachbandkabel.

# Das Gewebematrixtuch

Die Oberfläche eines Objektes mit Kontakten auszustatten ist notwendig. Denkbar ist hier eine Art Tuch mit das Objekt eingehüllt wird. Ein solches Tuch ist im Rahmen der Bachelorarbeit hergestellt worden.

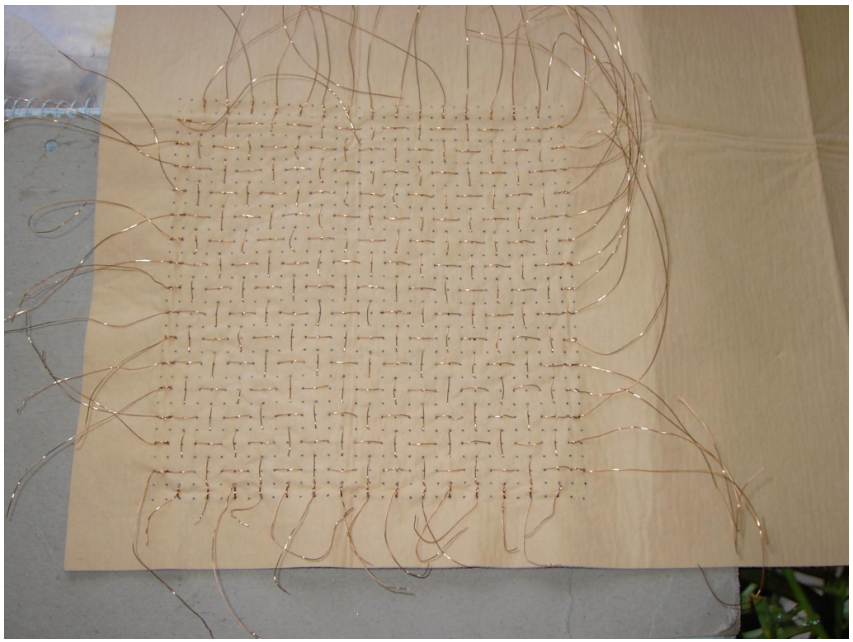
- Einfaches Fensterleder.
- Kontaktfläche von 16cm $\times$ 16cm.
- Mit 30 eingewebten Kupferlackdrähten.
- Entstehen 225 Kontaktkoordinaten.
- Anschluss über 34er Flachbandkabel.

# Das Gewebematrixtuch

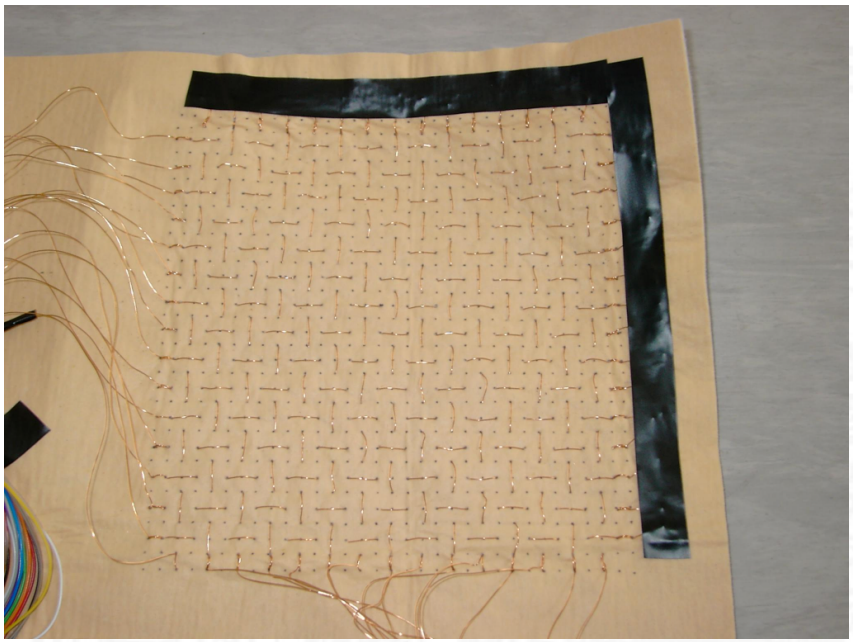
Die Oberfläche eines Objektes mit Kontakten auszustatten ist notwendig. Denkbar ist hier eine Art Tuch mit das Objekt eingehüllt wird. Ein solches Tuch ist im Rahmen der Bachelorarbeit hergestellt worden.

- Einfaches Fensterleder.
- Kontaktfläche von 16cm $\times$ 16cm.
- Mit 30 eingewebten Kupferlackdrähten.
- Entstehen 225 Kontaktkoordinaten.
- Anschluss über 34er Flachbandkabel.

# Das Gewebematrixtuch (Aufbau)

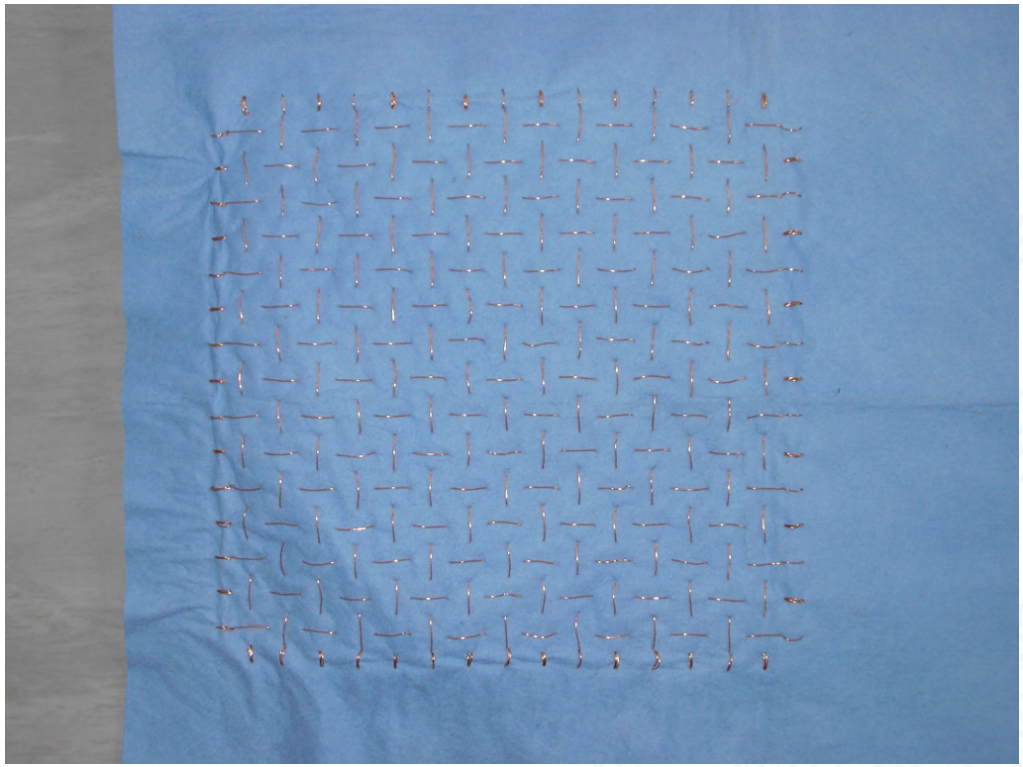


Kupferlackdrähte sind einzeln in das Leder gezogen.



Zusätzlichen Schutz vor ungewollter Kontaktierung bietet das Klebeband an den Enden.

# Das Gewebematrixtuch (Fertig)



Das fertige Gewebematrixtuch (Vorderseite) als SensorMatrix des ObjektControllers.

# Ausblick

Als realisierter Prototyp steht das taktile Sensorarray erst am Beginn der Entwicklung.
Dennoch sind die möglichen kommenden Entwicklungsschritte absehbar.
Diese weiteren Schritte können einerseits bei der Hardware aber auch bei der Software stattfinden.

# Ausblick der Hardware

Im Bereich der Hardware sind folgende Entwicklungen denkbar.

- Miniaturisierung
- TWI-Bus
- RS-232 → USB
- Kontaktmatrizen

# Ausblick der Hardware

Im Bereich der Hardware sind folgende Entwicklungen denkbar.

- Miniaturisierung
- TWI-Bus
- RS-232 → USB
- Kontaktmatrizen

# Ausblick der Hardware

Im Bereich der Hardware sind folgende Entwicklungen denkbar.

- Miniaturisierung
- TWI-Bus
- RS-232 $\rightarrow$ USB
- Kontaktmatrizen

# Ausblick der Hardware

Im Bereich der Hardware sind folgende Entwicklungen denkbar.

- Miniaturisierung
- TWI-Bus
- RS-232 $\rightarrow$ USB
- Kontaktmatrizen

# Ausblick der Software

Im Bereich der Software sind folgende Entwicklungen denbar.

- dynamische Adressierung
- Befehlsstruktur
- Filterung der Koordinaten
- kalibrierendes System

# Ausblick der Software

Im Bereich der Software sind folgende Entwicklungen denbar.

- dynamische Adressierung
- Befehlsstruktur
- Filterung der Koordinaten
- kalibrierendes System

# Ausblick der Software

Im Bereich der Software sind folgende Entwicklungen denbar.

- dynamische Adressierung
- Befehlsstruktur
- Filterung der Koordinaten
- kalibrierendes System

# Ausblick der Software

Im Bereich der Software sind folgende Entwicklungen denbar.

- dynamische Adressierung
- Befehlsstruktur
- Filterung der Koordinaten
- kalibrierendes System

# Demonstration des Prototypen

## Der funktioniert sogar!

Im Folgenden möchten wir Ihnen nun den, im Rahmen unserer Bachelorarbeit, entwickelten Prototypen eines taktilen Sensorarrays in seiner Funktion demonstrieren.

# Vielen Dank für Ihr Interesse!

**Nikolas Slottke**[1] **Hendrik Udo Linne**[2]
{7slottke, 7linne}@informatik.uni-hamburg.de

Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Department Informatik
**Technische Aspekte Multimodaler Systeme**