Karlsruhe Institute of Technology

Development of a Portable Measurement Device for Capacitor Arrays

Bachelor's Thesis
by

Tillmann Lange

at the
Department of Electrical Engineering and Information Technology
Light Technology Institute

Responsible Supervisor:	Prof. Dr. rer. nat. Uli Lemmer
Co-Supervisor:	Prof. Dr. Norbert Willenbacher
Advisor:	Dr. Hongye Sun

Project Period: 01/07/2020 – 31/12/2020

Karlsruhe Institute of Technology

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht und die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet habe.

Karlsruhe, den 31. Dezember 2020 _____

Abstract

The aim of this work was to develop a portable measurement device for flexible capacitor arrays that are called CapArrays in this work. CapArrays are 3D-printed capacitor arrays composed of materials such as Polydimethylsiloxane (PDMS) and Thermoplastic polyurethane (TPU) and conductors of the same materials infused with silver particles. This device was developed since no standard measurement device exist for measuring capacitance change of such arrays. A measurement method for distinguishing single points was presented. A device was developed and manufactured implementing this measurement principle. Additionally configuration and user interface software was written. Simple testing was successfully done to validate the functionality of the device. The CapArray meter is fully functional for it's intended purpose and presents a solid foundation for further research and development.

Contents

List of Figures

List of Tables

1. Introduction

The development of soft and flexible electronic devices opens up a lot of potential innovation in the fields of medical, industrial and consumer electronics. [1] presents a new way of infusing the Materials Polydimethylsiloxane (PDMS) and Thermoplastic polyurethane (TPU) with silver particles, retaining their mechanical qualities of stretchability and flexibility. These silver infused materials are also 3D printable via a Direct Ink Depositing technique, opening up the possibility of rapidly prototyping components. The KIT Institut für Mechanische Verfahrenstechnik und Mechanik (MVM) is currently developing multiple types of Sensors composed of these materials. Among them are capacitor arrays intended for multipoint pressure sensing. These are referred to as CapArrays in this work. The aim of this work is to develop a circuit, called the CapArray meter, for qualitative measurement of capacitance change of these arrays when subjected to pressure. The CapArray meter is intended to aid further development of these sensors and to enable exploring real world applications like pressure based touchpads or in-shoe pressure measurement for gait analysis.

2. Background & Related Work

2.1 Soft Matrices for Pressure Sensing

Soft, pressure sensitive capacitor arrays for pressure sensing composed of PDMS or TPU are currently in development at the MVM. One such array was provided for validation of the measurement principles and circuit presented in this work. Figure 2.1 shows this array.

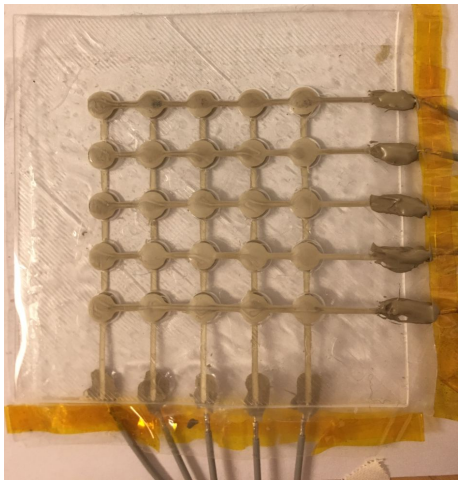

Figure 2.1: Printed PDMS soft sensor with glued in connection cables

These arrays are formed by overlaying two sheets of the base material with printed electrodes on top. These sheets have n rows of m interconnected electrically conductive printed dots. Before placing the sheets on top of each other and gluing them together one is rotated by 90 degrees, and aligned such that the dots of the top layer reside over those of the bottom layer. This forms $n \cdot m$ parallel plate capacitors with the printed dots representing the electrodes of these capacitors. The equivalent electrical circuit of a 3 by 3 array is shown in Figure 2.2.

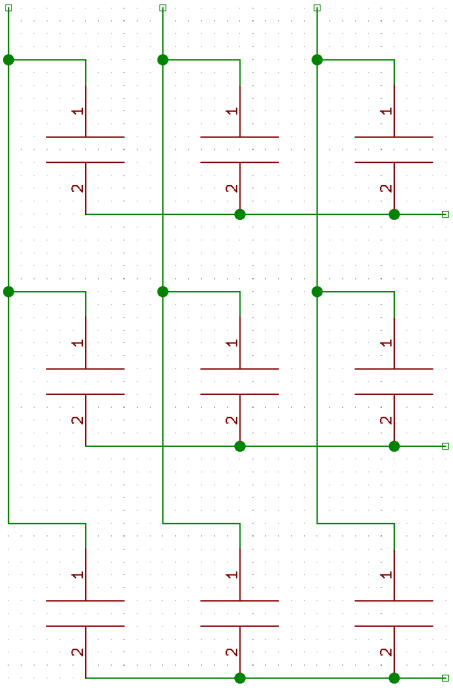

Figure 2.2: Equivalent circuit diagram of a 3x3 capacitive sensor array

As this construction results in n · m pressure sensitive points with only n + m outgoing connections it is highly scalable. Distinguishing between these points presents a major challenge and is discussed in Chapter 3. Figure 2.3 shows a cross section of a 3 by 3 array, similar to the aforementioned array provided for testing.

Figure 2.3: Cross section of a 3x3 soft sensor

Here an air hole is chosen as the dielectric to heighten pressure sensitivity as this array is intended for the pressure range exerted by a human finger. Note that the additional cast layers of material at the top and bottom of the sensor are only required for PDMS based sensors as the Ag-PDMS material is not stable when exposed to air. No such layer is required for TPU based sensors.

2.2 Capacitance Measurement

Capacitance can be measured in a number of ways suitable for low power applications [2] [3]. Such as

- measuring distortions in an AC signal (typically a square or sine wave)
- measuring the discharge time between two known voltage levels

- measuring the voltage drop when charged by a secondary capacitor with a known charge and capacitance
- measuring imbalance in a bridge circuit

The first two of these are what is typically implemented in the dedicated capacitance measurement ICs on the market. The PCAP04 chosen in 4.2.1 implements discharge measurement.

2.2.1 Discharge Measurement

Discharge measurement is a ratiometric approach to measuring capacitance. In this the time required to discharge a capacitor from one voltage to another via a known resistor is measured. (2.1) is the discharge equation for a capacitor C with R_C being the sum of the inner resistance of the capacitor and that of an external resistor used for discharging [4]. For a known $V_{th} = v_C$, V_0, t and R_C this solves to (2.2). Assuming the inner resistance of the capacitor and the series resistance of any wires is negligible compared to the external resistor this yields the capacitance. Figure 2.4 shows the discharge curve of a capacitor according to (2.1) with the time constant $\tau = R_C \cdot C$. The threshold voltage V_{th} should be chosen in the upper almost linear region of this curve.

$$v_C(t) = V_0 \cdot e^{-\frac{t}{R_C \cdot C}} \tag{2.1}$$

$$C = \frac{t}{R_C} \cdot \frac{1}{\ln \frac{V_0}{V_{th}}} \tag{2.2}$$

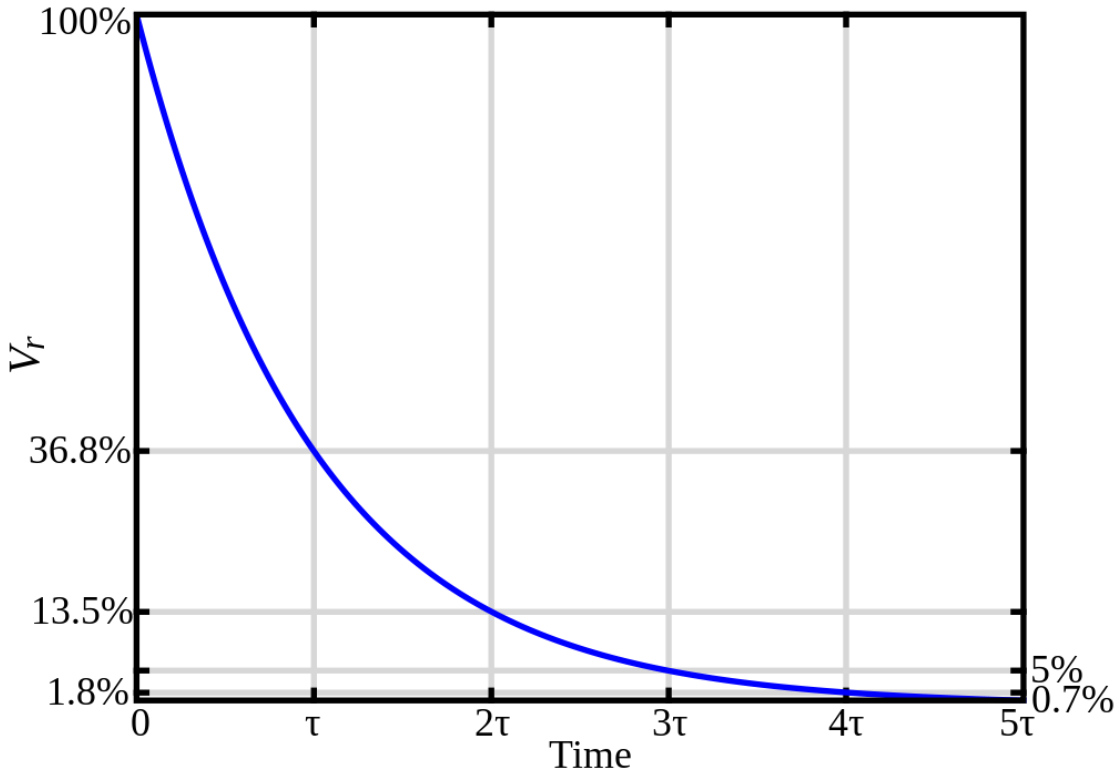

Figure 2.4: Discharge curve of a capacitor. From [5]

2.3 Sensitivity, Accuracy and Precision

The terms sensitivity, accuracy and precision are common scientific parlance. They are defined in this section to clarify what they mean for the purposes of this work.

Sensitivity or measurement resolution is the smallest change of a physical property that can be measured. Sensitivity to capacitance change is the primary focus of this work.

Precision is the degree to which two measurements under unchanged conditions produce the same result. Precision can be gained by increasing the sample size.

Accuracy is the closeness of a measurement to an absolute standard, in this case the capacitance in Farad or the pressure in Pascal. Establishing accuracy in either of those dimensions is outside of the scope of this work. Accuracy of the capacitance measurement would require additional characterisation of the circuit in regards to stray capacitance and additionally temperature compensation. Accuracy of pressure measurement would require characterization of the CapArray.

2.4 Capacitance for printed arrays

The equation for the capacitance of a parallel plate capacitor is given in (2.3). Under an external pressure the distance between the electrodes decreases and the capacitance changes. This change in capacitance is the variable to be measured by the CapArray meter.

$$C = \varepsilon_0 \varepsilon_\mathrm{r} \cdot \frac{A}{d} \tag{2.3}$$

The array shown in Figure 2.1 has an electrode radius of 2.5 mm and an initial distance of 1 mm. Under pressure this distance can be reduced to around 0.2 mm. The resultant capacitance is around 0.2 pF in the relaxed state and 1 pF when fully pressed.

The designer of a CapArray has a number of options for reaching a desired capacitance change such as

- changing the electrode size
- using another material with a different ε_r
- changing the initial distance of the electrodes

Each of these has their own limitations and difficulties and also changes the pressure response of the array.

2.5 Pressure Ranges for Proposed Applications

Currently two possible applications for the CapArrays have been proposed, the use as a pressure sensitive multi touch touchscreen and for measurement of the pressure at the sole of the foot for gait analysis. These are 10 kPa - 40 kPa for human finger touch and >1 MPa for in-shoe pressure according to [6]. This work focuses on achieving adequate sensitivity for touch sensing. As higher pressures also naturally cause higher changes in capacitance in-shoe pressure sensing is deemed less problematic. Therefore it is assumed that a device that can sense finger touch can also be used for in-shoe pressure with a differently tuned CapArray.

2.6 Parasitic Capacitance

Parasitic or stray capacitance is an unwanted capacitance that exists between elements of a circuit because of their closeness. In discharge time measurement of capacitors this capacitance does not affect sensitivity. Any static stray capacitance in the measurement path is simply a static offset on the measured result. It does however have an impact on the time required to charge or discharge the measurement path.

For the CapArray meter connected to the CapArray shown in 2.1 parasitic capacitance is estimated at about 60 pF while the capacitance range to be measured is $< 1pF$. Another problem is stray capacitance between the CapArray or the CapArray meter with objects close to them. For example the desk they are placed on or a human finger touching them. As this stray capacitance often varies over time it introduces an error to the measurement. This can be avoided by shielding all conductors with a metallic insulation layer. How to prevent this insulation layer itself from introducing a large parasitic capacitance is described in Section 3.2. Shielding and suppression of the associated parasitic capacitance is already implemented for the coaxial cables connecting the CapArray to the CapArray meter.

3. Analysis

3.1 Measurement Concept

A major challenge in measuring these CapArrays is distinguishing the capacitance change of a singular capacitor rather than that of multiple.

3.1.1 Simple Measurement without Compensation

The problem is illustrated in Figure 3.1a, showing a 3x3 array that is measured with reference to ground.

Through switches the capacitor C1 is selected for measurement in a simple matrix addressing scheme.

The equivalent circuit in Figure 3.1b shows why this can not work. Besides the intended ground path of C1 a number of different paths over the other capacitors also exist. As long as C2 - C7 remain static they present in the measurement as a static offset. If multiple capacitors are being pressed the measured result does no longer reflect the pressure applied to C1.

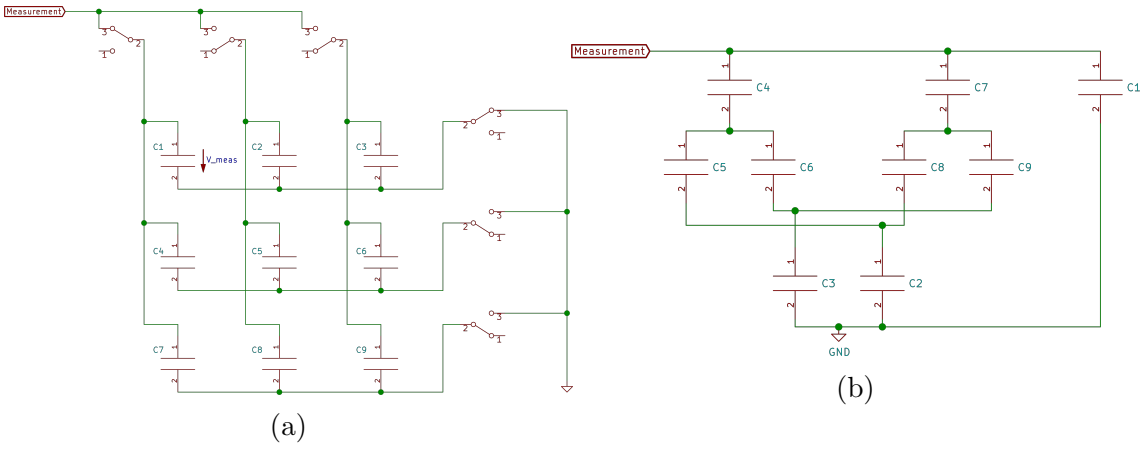

Figure 3.1: Equivalent circuit diagram of a 3x3 sensor array without compensation

3.1.2 Crosstalk Compensation via V_{Guard}

Figure 3.2 shows a solution to this problem. In this figure only C1, for which measurement is desired, is being referenced to Ground. The other two capacitors that are connected to the measurement path, C4 and C7, are connected to the stable reference voltage V_{Guard}, instead of being left floating as in Figure 3.1a. V_{Guard} is a special Op-Amp driven reference voltage provided by the PCAP04. It mirrors the voltage V_{meas} being used as charging voltage for the measurement. Since $V_{meas} - V_{Guard} \approx 0$ only C1 is charged by the voltage V_{meas} provided by the measurement device. Thus C4 and C7 are not charged. The other paths to ground as shown in Figure 3.1b no longer exist in this configuration.
C2 and C3 are charged, but don't interact with the measurement path. C5, C6, C8 and C9 are shorted to V_{Guard} and thus not charged.

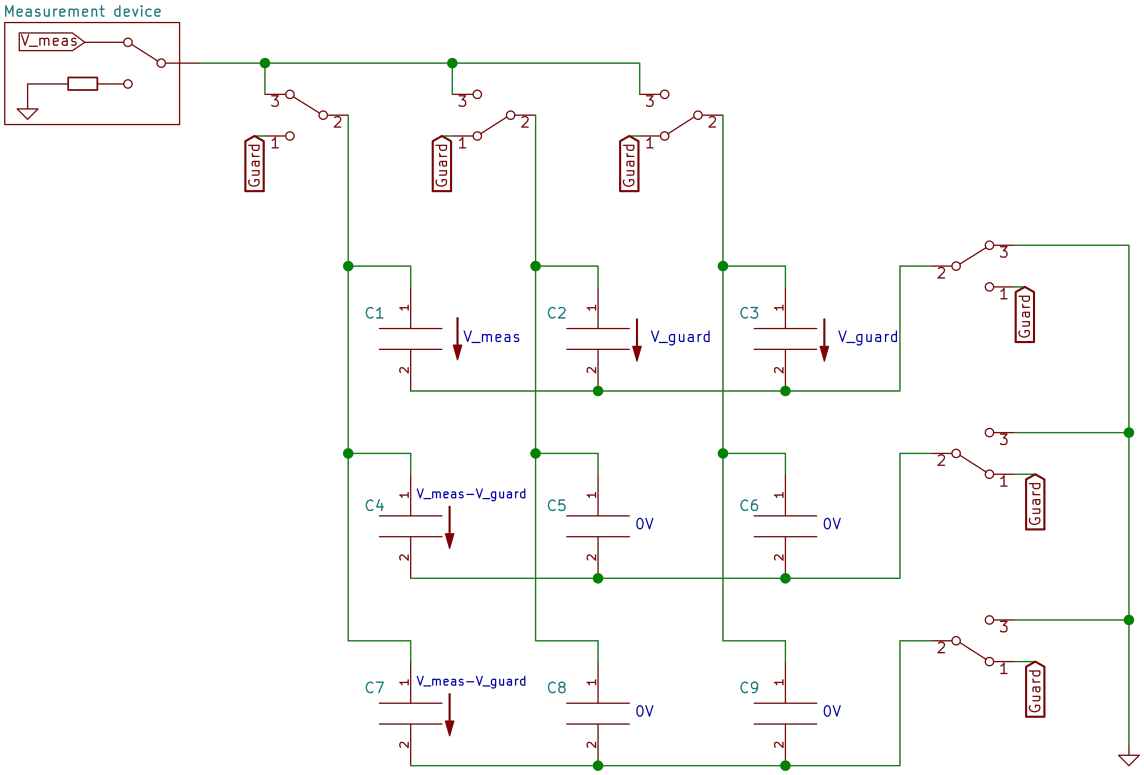

Figure 3.2: Equivalent circuit diagram of a 3x3 sensor array using V_{Guard} for crosstalk compensation

In the second stage as shown by Figure 3.3 the measurement device switches internally to a discharge resistor connected to ground. Measuring the time until the voltage $V_{C1}(t)$ drops to a threshold voltage V_{th} yields the capacitance of C1 as shown in Equation (2.2).

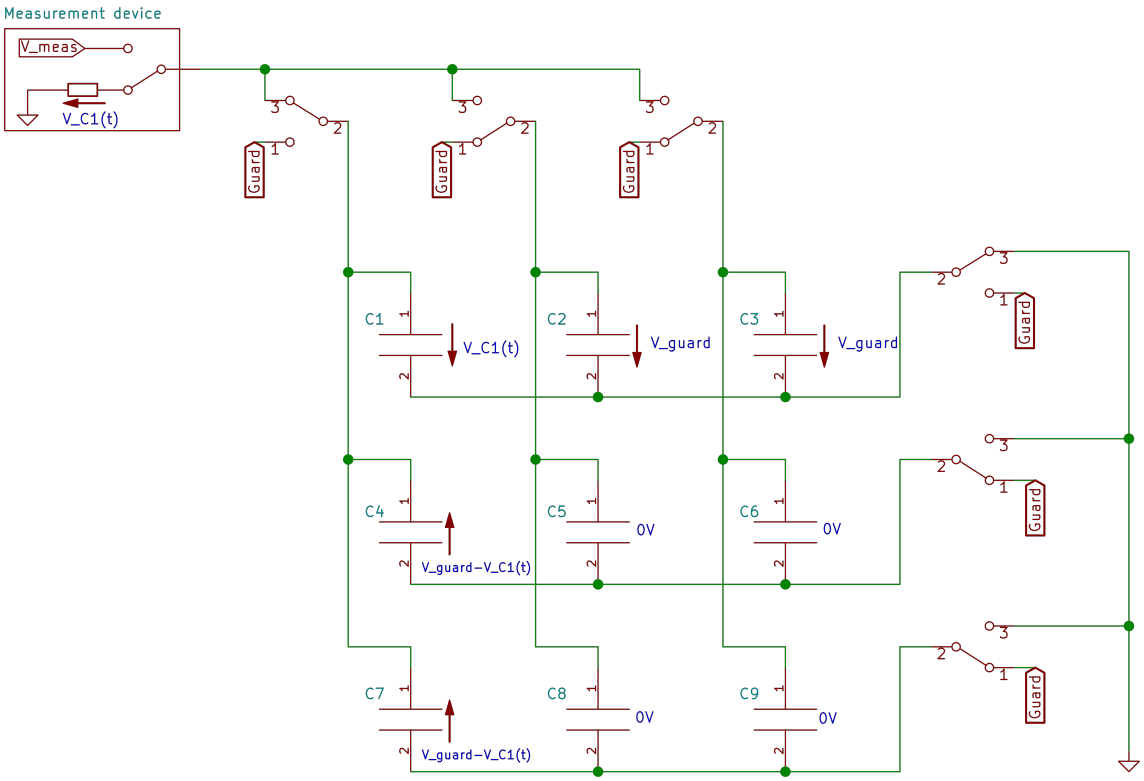

Figure 3.3: Equivalent circuit diagram with voltages during discharge measurement of C1

During this time V_{Guard} follows $V_{C1}(t)$. C4 and C7 remain uncharged. Figure 3.4 shows one charge-discharge cycle measured with an Agilent DSO9254A 2.5GHz oscilloscope using Keysight N2843A probes with 10 MΩ probe impedance and 11 pF parasitic probe capacitance. The green waveform is that of the measurement path, the blue one that of V_{Guard}. The measurement treshold voltage V_{th} is not specified in the PCAP04 datasheet but is assumed to be somewhere within the light blue markers. In this region the matching between V_{Guard} and $V_{C1}(t)$ is nearly perfect. Figure 3.5 shows this in more detail. The yellow waveform is a differential measurement of V_{Guard} and $V_{C1}(t)$, taken with a Agilent N2750A differential probe with 200 kΩ probe impedance and 700 fF parasitic probe capacitance. The blue waveform is V_{Guard} as a visual reference. At around 1V measurement voltage a small mismatch between V_{Guard} and $V_{C1}(t)$ appears. This may be caused by the current draw of the PCAP04s internal CPU being triggered or by some internal switching mechanism after V_{th} is reached.

Figure 3.4: Waveforms of $V_{C1}(t)$ (green) and V_{Guard} (blue)

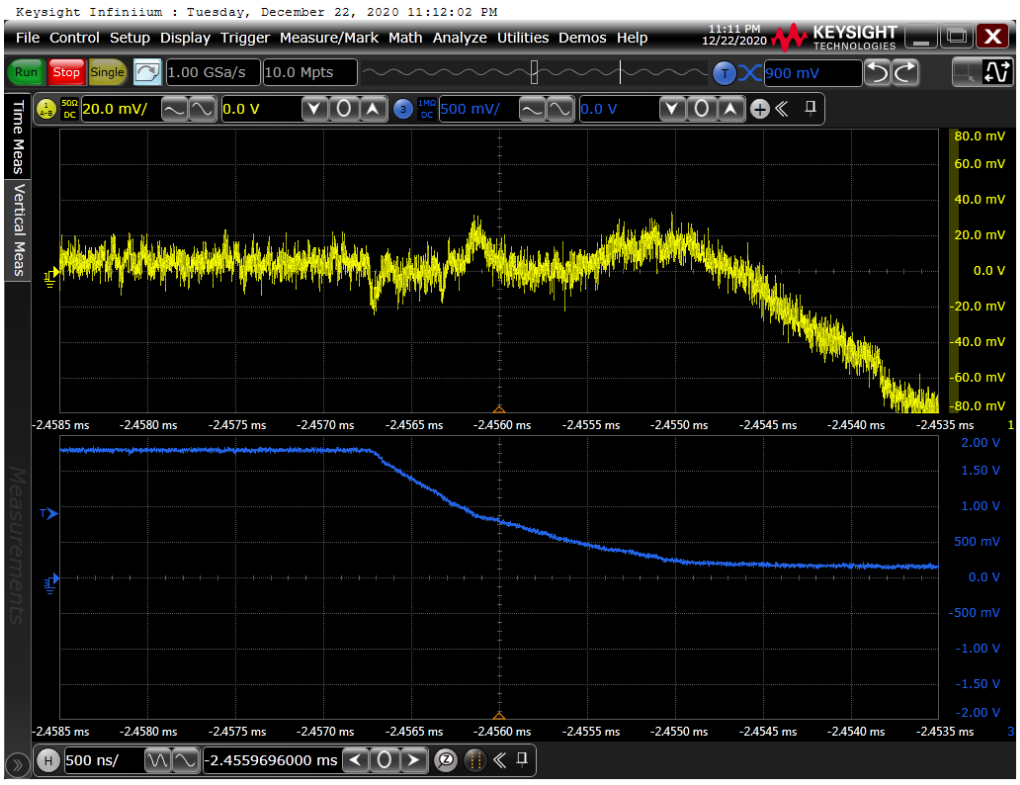

Figure 3.5: Waveform of the difference between $V_{C1}(t)$ and V_{Guard} (yellow). V_{Guard} (blue) is shown as a time reference

To prepare for the measurement of another point all switches are pulled to Guard, shorting the capacitors and dissipating any remaining charge.

3.2 Parasitic Capacitance

Figure 3.6 shows the parasitic capacitance in the measurement path in an exemplary way. C'_{cdc} is the parasitic capacitance between the measurement IC and the ground plane. C'_{mux} is the parasitic capacitance of the switch multiplexer. This capacitance is assumed to have the same value for all switch positions used by the CapArray meter. This is a simplification as there are likely internal differences between the multiplexers ports. The inputs 1 and 3 are common to each of the multiplexers switches. This presents an advantage over using individual switches where the inputs of each switch add an additional parasitic capacitance. C'_{bt} is the parasitic capacitance between the board traces and the ground plane. For the PCB stackup used in Chapter 4 this is estimated at about $\frac{1pF}{cm}$. These traces have been routed to be as short as possible.

For different port combinations different lengths of trace are part of the measurement. For the most disadvantageous combination of ports this length is around 3.5 cm.

The most problematic parasitic capacitance has already been avoided in Figure 3.6 by pulling the outer conductor of the coaxial cables connecting the circuit to the CapArray to V_{Guard}. The parasitic capacitance of 1.32 mm coaxial wire is typically about $\frac{1pF}{cm}$ [7]. Getting rid of the influence of wire length enables convenient placement of the CapArray meter.

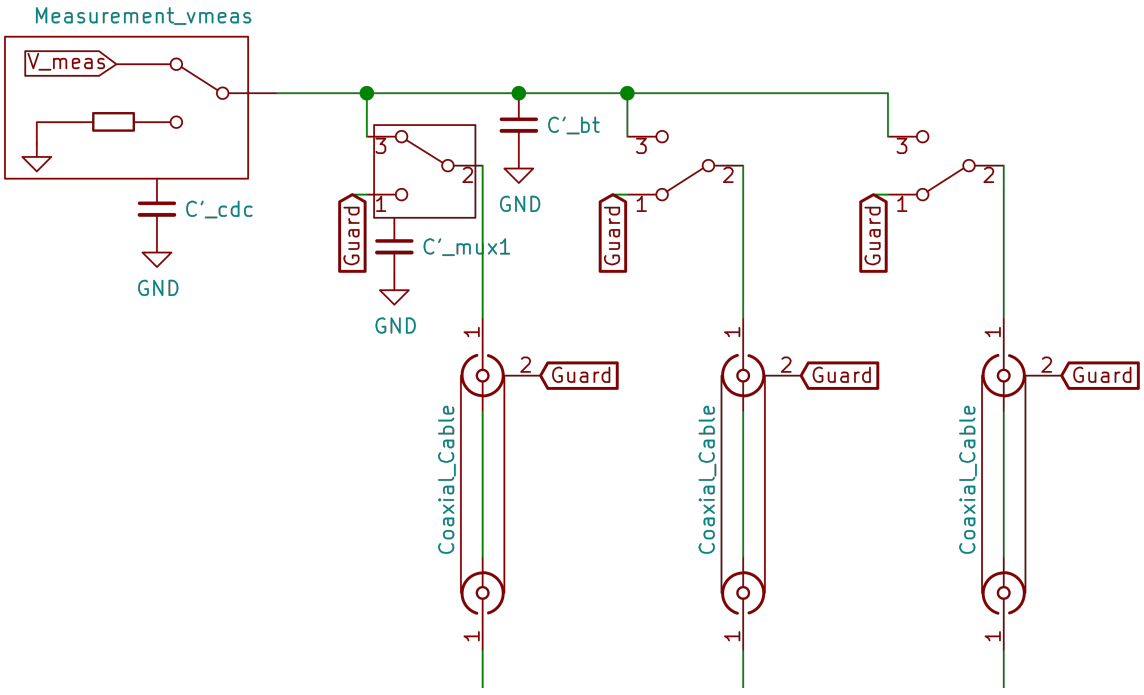

Figure 3.6: Parasitic capacitance of the measurement path

4. Hardware Development

4.1 Design goals

The goal of this work was to obtain a small and versatile measurement circuit ready for testing different use cases of the sensor arrays rather than cumbersome benchtop devices that constrain testing to laboratory environments.

4.1.1 Size

The size proposed for this device is roughly that of a matchbox (50 mm by 35 mm). This is small enough to carry it in a pocket, strap it to a wrist or ankle or sew it into clothing or shoes for testing purposes. No special considerations were taken for height as SMD components typically aren't problematic in this regard. The dimensions of the assembled CapArray meter are 43 mm x 35.4 mm x 5 mm.

4.1.2 Sensitivity and Precision

No formal requirement for sensitivity or precision was made. The benchmark used during development was for the CapArray meter to be precise and sensitive enough to detect the pressure exerted by the touch of a human finger on the CapArray provided for testing purposes.

4.1.3 Environment Protection

This device and the sensor array are supposed to be able to be carried closed to the human body. Thus connections between them need to be protected from parasitic capacitance.

Temperature is another environmental factor to be considered. No temperature compensation was done for this work, the resultant circuit is supposed to be used at room or skin temperature.

4.1.4 Power Consumption

No special regard has been given to power consumption. All components used for this circuit can be classified as 'low power' however. Optimising it for battery use should not prove hard. Using the configuration detailed in 5.5 the current consumption of the CapArray meter is around 70 mA.

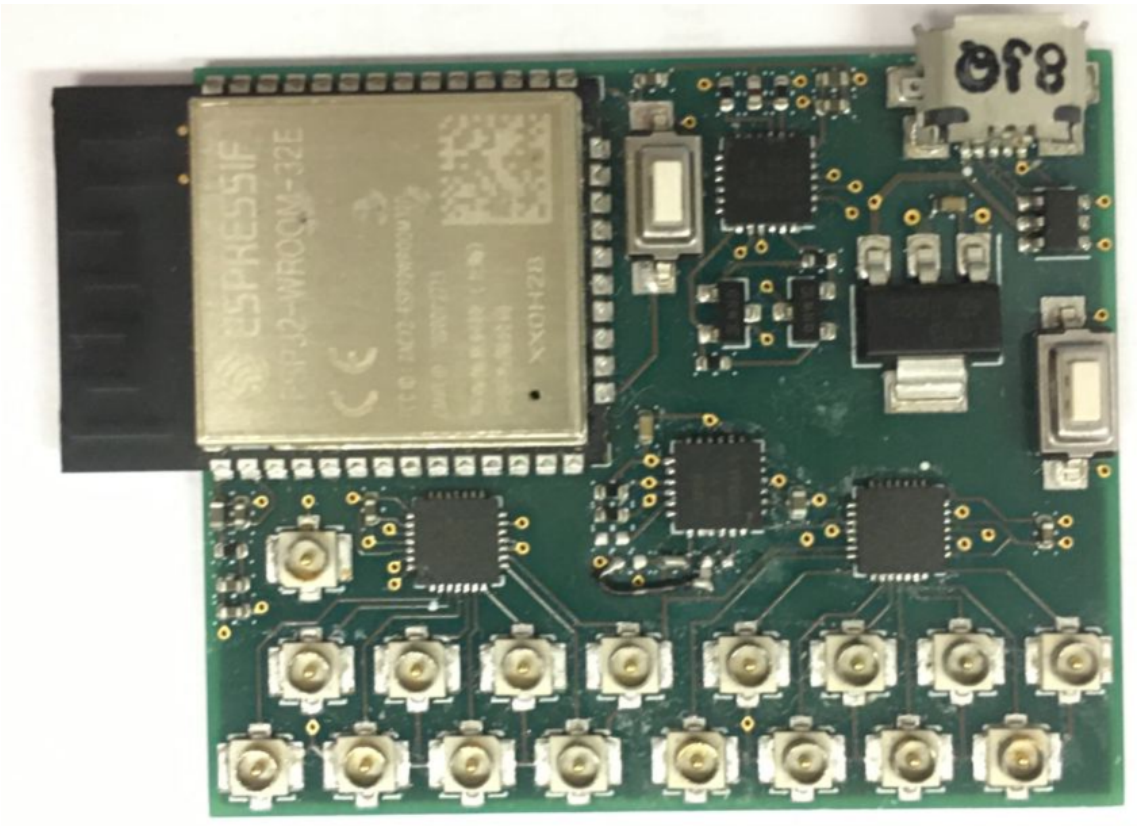

Figure 4.1: Picture of the second revision of the CapArray meter

4.1.5 Ease of Use

Using this device should not require uncommon hard- or software or special knowledge not obtained reading this thesis.

4.1.6 Ease of Further Development

In future use cases of the soft sensor arrays the requirements for the measurement circuit may change. The design decisions made for this device should facilitate further development and enable further customization of the circuit and software applications.

4.2 Selection of Components

To achieve these aims appropriate components were chosen. A focus was placed on selecting small and low powered components. All components are useable at a Vdd of 3.3 V.

The package size 0402 is commonly used on this board. This size was chosen since all resistors and capacitors needed for this circuit are available in this form factor.

4.2.1 Capacitance to Digital Converter

A number of specialised Capacitance to Digital Converter (CDC) integrated circuits exist on the market today. The CapArray meter is based on the state of the art PCAP04 [3], released in 2017, for its wide range of application.

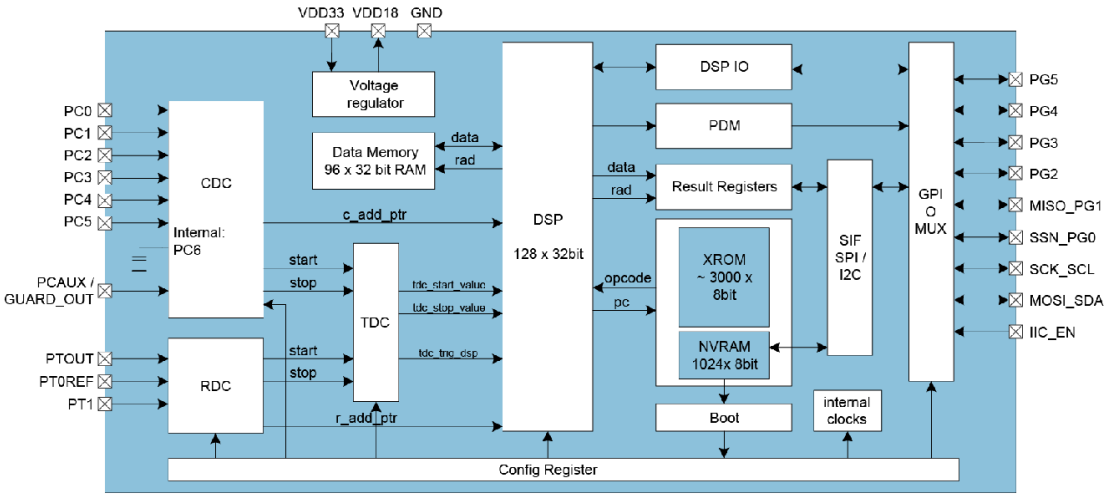

Figure 4.2: Block diagram of the PCAP04 as shown in the datasheet [3]

Due to its 20-bit maximum effective measurement resolution it is one of the most precise devices on the market. Power consumption can be as low as 3 µA. It supports high sampling rates of up to 50 kHz and a has a measurement range of 1 pF to 100 nF.

For each of these characteristics individually other ICs offer similar performance, but the range of customizability of the PCAP04 is outstanding.

Additionaly the PCAP04 also has an internal 32-bit Harvard architecture CPU which can run user defined code and communicate via dedicated IO pins.

This CPU could in theory be used to control switches directly. The feasibility of this was not explored since the primary purpose of this CPU is signal processing. The PCAP04 also supplies the voltage V_{Guard} mentioned in 3. This simplifies the design process as no additional Op-Amp is needed to supply such a voltage.

4.2.2 Choice of Bus

The PCAP04 offers a choice of either the I2C or SPI serial interface. For both only the slave role is supported. I2C was used because it requires less communication lines than SPI.

Due to a design flaw of the PCAP04 described in Errata sheet [8] I2C should only be used for point to point communication. This flaw was not considered during the hardware design phase and I2C is used as a bus. The ESP32 acts as bus master while the PCAP04 and the analog switch multiplexers act as slaves. Due to the design flaw some I2C commands intended to control the switch multiplexers are

interpreted by the PCAP04 as valid commands. A workaround based on specific command sequences is described in Section 5.1.

4.2.3 Microcontroller Unit

There are tasks the MCU is required to perform, namely switching the analog switches, flashing configuration data to the PCAP04 and capturing measurement data from the PCAP04. None of these tasks are particularly resource intensive. Communication via I2C bus is a standard feature for microcontrollers.

The ESP32-WROOM [9] module was selected for ease of use. It is oversized in regards to performance and memory and could easily be swapped out for a smaller MCU that consumes less power and board space. The ESP32-WROOMs capability of communicating via WiFi, Bluetooth and Bluetooth low Energy could be of interest for future development.

4.2.4 Connectivity

USB support is a stable and ubiquitous option for programming and experimentation tasks in combination with a personal computer. Therefore an USB to UART converter is included on the board to connect the incoming USB signal to the MCU. USB also provides a 5V rail used to power the device and a GND connection that is usually connected to Earth Ground (if possible) at the host side. Through this only an USB connection and no other peripherals or converters are needed to operate the CapArray meter. A Micro USB port is implemented for its small size. The CP2102N [10] USB to UART converter is chosen since it is recommended for the ESP32 MCU.

4.2.5 Development Environment

The Arduino Environment abstracts many of the lower level functions of a microcontroller and as such saves time and effort in the development of code.

The ESP32 MCU is widely used and well supported on the Arduino development platform.

4.2.6 Measurement Path Switching

For the adressing scheme presented in Chapter 3 each outgoing port needs it's own analog switch. In the first board revision individually packaged analog switches were used. The MCUs GPIOs were used to provide individual control signals. This approach proved to be space consuming and is also limited by the number of free GPIO lanes of the MCU (16 in the case of the ESP32-WROOM module). The second revision replaces these individual switches with multiplexers to save board space. This was required anyhow since the concurrent development of the sensor arrays progressed to a point where testing of up to 8x8 pressure points was desirable.

Since there aren't many multiplexers on the market that switch one of two signals to one of multiple (8 and upwards) ports the choice fell to the I2C addressable MAX14661 [11] 16:2 multiplexer. The functional diagram of the MAX14661 is shown in Figure 4.3. Two such multiplexers are required for the addressing scheme presented in Chapter 3.

The only additional components required for an upgrade to 16x16 points are additional coaxial connectors.

I2C addressing also removes the dependency on the number of available GPIO lanes of the MCU.

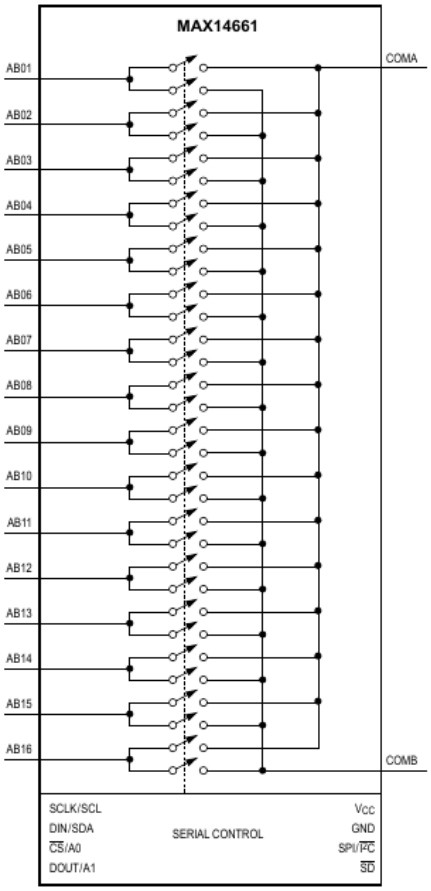

Figure 4.3: The functional diagram as shown in the MAX14661 datasheet [11]

4.2.7 Sensor Ports

Coaxial cabling is desirable since it provides shielding to protect the measurement path from outside influences and its parasitic capacitance can be eliminated as explained in Section 3.2.

U.FL compatible coaxial cables and connectors are used for this because of their small board footprint and mechanically stable connection.

4.3 PCB Design

The circuit can be divided into a number of subsystems that interact with each other, as shown in Figure 4.4. A schematic and PCB layout implementing these subsystems were designed in KiCad. The schematic is included in Appendix B.

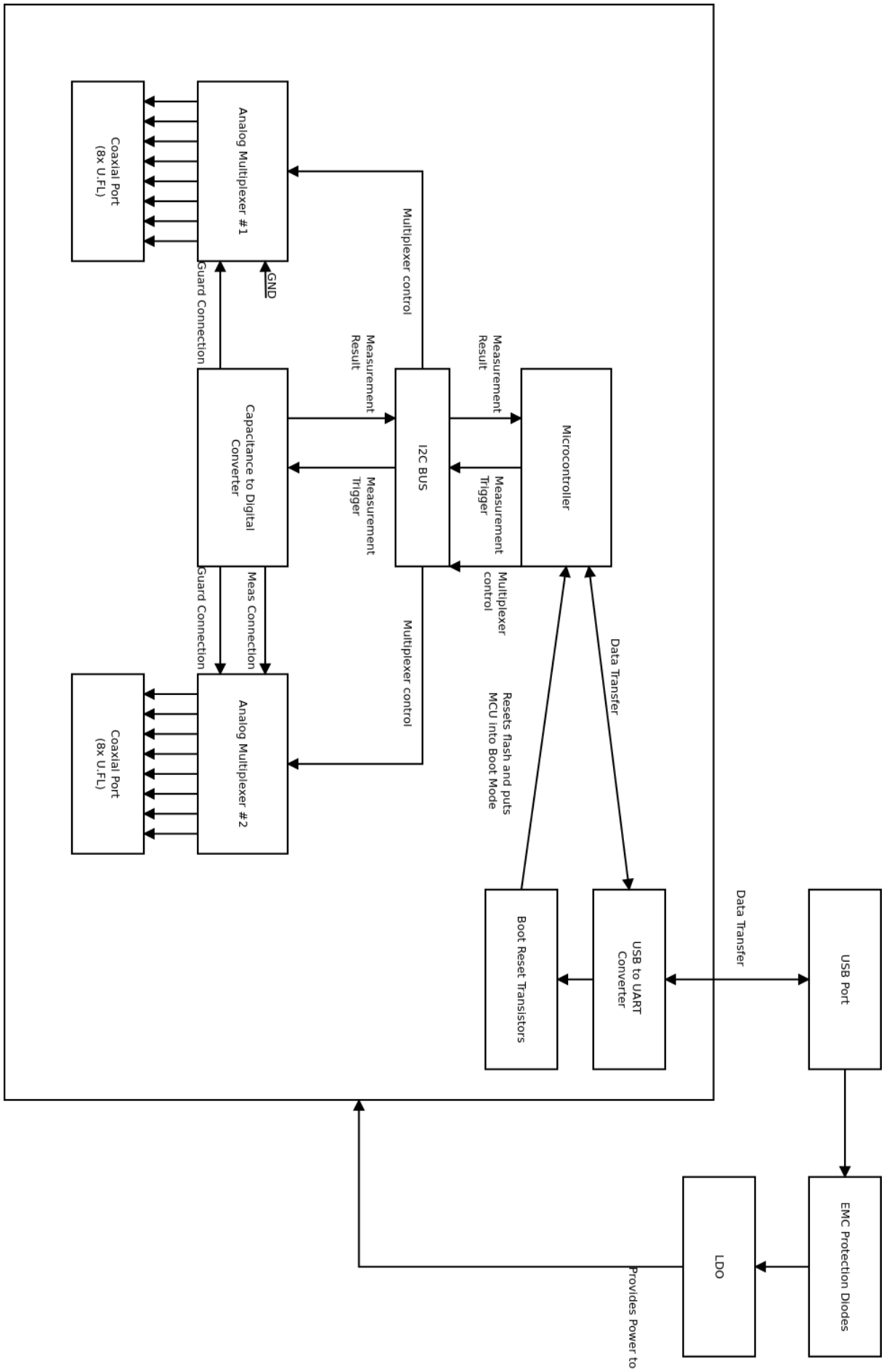

Figure 4.4: Block diagram of the CapArray meter

4.3.1 Power

The power circuit for the CapArray meter is a standard approach. 5 V are provided by the Micro-USB port in accordance to the USB specification [12] and subsequently regulated down to 3.3 V by a SOT-223 LT1117-3.3 [13]. The 3.3 V power rail is used as V_{dd} by all other ICs on the board. The devices share GND with the USB connection. ESD protection is provided by an USBLC6-2SC6 [14] protection diode.

4.3.2 Serial Communication

For access to the MCU a CP2102N [10] USB to UART convertor is used, additionally two bipolar transistors are required to enable programming the integrated FLASH memory of the MCU. The TXD pin of the CP2102N needs to be connected to the RXD0 pin of the ESP32-WROOM and RXD to TXD0.

4.3.3 Reset and Boot Reset Buttons

Push button switches on EN and IO0 provide reset and boot reset functionality.

4.3.4 I2C Bus

The CapArray meter uses the I2C protocol for all communication between the MCU and the measurement subsystem. This only requires two additional 4.7 kΩ pullup resistors on the SDA and SCL lines connected to IO21 and IO22 of the MCU. These lines are then directly connected to the respective SDA and SCL pins of the CDC and switch multiplexers.

4.3.5 Measurement

The measurement subsection of the device consists of the PCAP04 in a QFN-24 package, some associated decoupling capacitors, a C0G/NP0 10 pF reference capacitor, two MAX14661 [11] multiplexers and two banks of eight U.FL compatible male connectors.

A guard distance is implemented between the rest of the circuit and those parts relevant for the capacitance measurement, in particular between the PCAP04 and the digital circuitry used for wireless communication.

All components reside over a Ground plane on layer two of the PCB. Measures such as star grounding or pulling the plane under the measurement circuit to V_{guard} were foregone in favor of this simpler design. No special measures have been taken to isolate board traces carrying the measurement path against outside interference. These traces are however routed to be as short as possible.

4.4 Manufacturing

The PCB was manufactured by Würth Elektronik GmbH & Co. KG.

Components and solder paste were placed by hand and subsequently soldered using a vapor phase process.

A slight change from the manufactured layout was introduced. The reference capacitor C_ref1 was connected to GND instead of the PCAP04 port PC1. One side of C_ref1 is connected to the originally intended soldering pad, while the other side is connected to ground via a short jumper wire.

5. Software Development

5.1 I2C

All communication between the MCU and the measurement subsystem of the circuit is done via the I2C bus protocol. The Arduino IDE abstracts this protocol in the Wire library. To connect to a desired device one simply calls Wire.beginTransmission() to transmit or Wire.requestFrom() to receive data with the desired 7-Bit I2C address. The data is then placed into a transmission buffer with Wire.write() and written out via Wire.endTransmission() or read from the receive buffer with Wire.read(). Care has to be taken with I2C transmissions on this device as there is an design flaw within the PCAP04 that makes it listen to all I2C transmissions regardless of the submitted address. This causes unpredictable behaviour since any data sent on another device on the I2C bus might be recognised by the PCAP04 as a valid command. This was published in the Errata sheet [8]. Another revision of the circuit board should include hardware changes to account for this.

Two workarounds are included in this software to keep the device functional. The command normally issued for certain switch combinations of the MAX14661 causes the PCAP04 to crash until power cycled. This was patched by writing to a different register of the MAX14661 called the control register. The control register is a special register used to switch only one specific switch.

The second workaround is setting the PCAP04 to 'Opcode triggered' mode instead of 'Read triggered' mode since communicating with the multiplexers seemed to randomly trigger measurements in 'Read triggered' mode.

5.2 I2C Flashing of the PCAP04

The PCAP04 is operated in what the datasheet calls 'Pure Slave' mode.

Unlike described in the datasheet firmware persists between power cycles by default. Thus the firmware included with the manufactures development software is only written once, rather than after each power cycle.

A standalone mode using non volatile memory also exists but is not particularly useful for this project.

Configuration Registers are written each power cycle. This is implemented in the file main.ino shown in Appendix A in the lines 10 - 69. The configuration used is detailed in Section 5.5.

5.3 Measurement and Readout Sequence

To measure one specific point in the sensor matrix the appropriate switches of the two multiplexers are selected via I2C. This is done by writing the desired state of each individual switch to the MAX14661's switch register. This method has to be avoided when turning the switches AB14 and AB06 to ON (PC2 or GND respectively) and all other switches to OFF (Guard) as sending the register state 0x20 locks up the PCAP04.

After this a measurement is triggered by sending the trigger Opcode 0x8C to the PCAP04. After waiting long enough for the measurement to conclude the results can be read from the CDCs result register.

5.4 Data Processing

The PCAP04 uses a 32-bit unsigned fixed point number for a single measurement result. 5 bits are devoted to the integer part of the result, the remaining 27 bits to the fractional part. This is converted by the application running on the MCU to a 32-bit floating point value.

5.5 Configuration of the PCAP04

The PCAP04 measurement frontend is configured via 64 8-bit registers, these are listed in [3, p. 19–21].

Some registers are mentioned multiple times in tables 5.1 - 5.4 as they contain more than one configuration option. For finding out specifically which bits of these registers are responsible for which option the datasheet should be consulted.

5.5.1 General Settings

Register	Value (Hexadecimal)	Description
47	0x01	Runbit, enables measurement and DSP
28	0x5A	Disables the watchdog, recommended for operation as slave
13	0x18	Sets the CDC to 'Opcode triggered' mode

Table 5.1: General settings registers

5.5.2 Measurement Settings

This configuration triggers one measurement of the port PC2 against ground and one measurement of the reference capacitor at PC0 against ground. Each of these

measurements is composed of two 'fake' and 98 real samples that are averaged before the measurement result is written to the result register.

Fake measurements are conducted like real measurements but the results are discarded and do not reflect on the final result. This is done to dissipate any remaining charge in order to ensure that the device under test is always in the same state when measurement begins.

Register	Value (Hexadecimal)	Description
7	0x05	Enables PC2 (measurement port) and PC0 (reference measurement port)
2	0x68	Sets the discharge resistance $R_{discharge}$ to 30 kΩ
3	0x10	Sets charging resistance to 10 kΩ
4	0x00	Specifies that the measurement is made on a single channel against a Ground reference. Does not enable any of the built in compensation options.
15	0x04	Triggers one 'fake' measurement to reset the capacitor to a known state. The results of this measurement are discarded.
7 8	0x62 0x00	Number of samples to be averaged before writing results, up to 8191. 0x62 equals 98 samples

Table 5.2: Measurement setup registers

A measurement of the reference capacitor is mandatory and can not be skipped. Figure 5.1 shows the order of events for one measurement as shown in the PCAP04 datasheet.

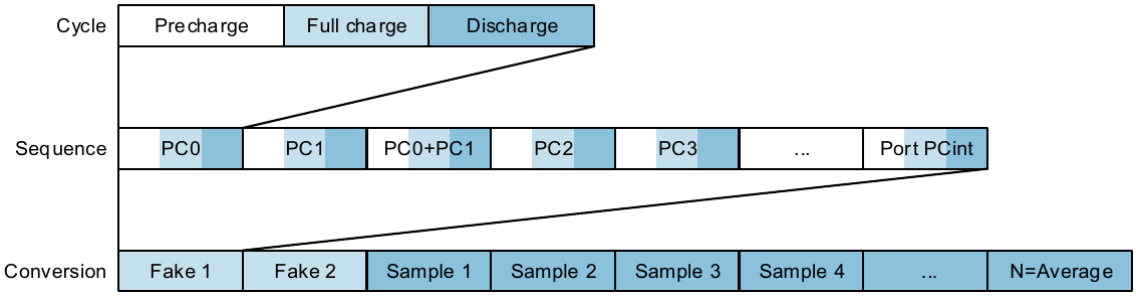

Figure 5.1: Order of events during a measurement as shown in the PCAP04 datasheet [3]. In the configuration shown in this chapter only the measurements on PC0 and PC2 are conducted in the 'Sequence' column

5.5.3 Cycle Time Settings

This section configures the length of the charge and discharge cycle. The clock source of the CDC has the frequency $\frac{1}{4} \cdot 2MHz$. Therefore the base time T_{cycle} is 2 μs. Precharge and fullcharge time together have to be long enough to fully charge the capacitor and stray capacitance in the circuit. With a base time of 2 μs writing the minimum value of 1 to these registers is sufficient. $T_{discharge}$ needs to be long

enough to fully discharge the circuit. In the time $5 \cdot R_{discharge} \cdot C$ approximately 99% of the charge is dissipated. Since the parasitic capacitance of the circuit is about 60 pF a time $> 9\mu s$ needs to be chosen.

Register	Value (Hexadecimal)	Description
1	0x01	Enables 2MHz high frequency clock
5	0x08	Sets HF clock as clock source for the CDC, divides clock frequency by 4 resulting in $T_{cycle} = 2\mu s$
14 15(bit 0:1)	0x01 0x05	Sets $T_{precharge} = T_{cycle} * (x + 2) = 6\mu s$
16 17(bit 0:1)	0x01 0x00	Sets $T_{fullcharge} = T_{cycle} * (x + 2) = 6\mu s$
12 13(bit 0:1)	0x06 0x18	Sets $T_{discharge} = T_{cycle} * x = 12\mu s$
30	0x81	Sets DSP trigger to end of CDC conversion
27	0x00	Sets DSP to fastest speed

Table 5.3: Measurement cycle time registers

5.5.4 Guarding

To use the measurement principle shown in Chapter 3 the Op-Amp driving the reference voltage V_{Guard} needs to be enabled.

Register	Value (Hexadecimal)	Description
1	0x01	Enables the high frequency clock, mandatory when guarding is used
5	0x08	Sets HF clock as clock source for the CDC, also mandatory
18	0x3F	Enables guard on all CDC ports that are not currently being sampled. Guard is only driven high during measurement.
20	0x07	Maximum driving strength for the guard Operational Amplifier

Table 5.4: Guard voltage registers

5.6 User Interface

The CapArray meter uses the Arduino IDE as its user interface. Measurements are requested from the CDC through the function enablePoint(int left, int right) defined in the file readoutroutine.h documented in Appendix A. Figure 5.2 shows the numbering of ports as used for this function.

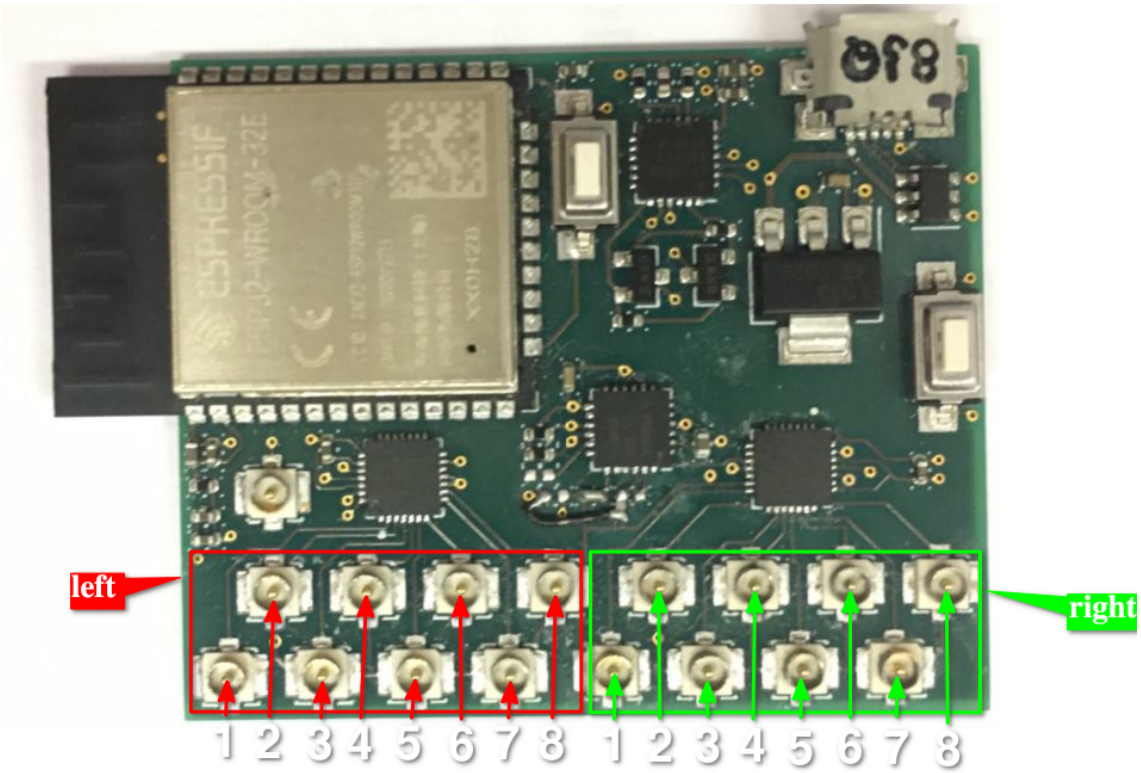

Figure 5.2: Sensor port numbering scheme used for enablePoint()

The function enablePoint() returns the value of the PCAP04s result register for the targeted point as a floating point number. The main routine of the file main.ino documented in Appendix A calls this function for all selected points and writes the result, as well as a reference to which point was pressed, to the PC via the serial interface. This serial message is formatted such that the 'Serial Plotter' program included with the Arduino IDE shows the measurements as differently colored lines. The plots are annotated as AA...DD in standard matrix notation. Letters are used instead of the more usual numbers to avoid the Serial Plotter from plotting them as a line instead of using them as labels. main.ino is configured to read out a 5x5 array. Smaller sections or single points of the array can be plotted by adjusting the bounds of the for loops on line 73 and line 74.

This user interface allows access to all relevant configuration.

The Arduino IDE is available for all major operating systems, making this user interface platform independent.

6. Evaluation

To validate the measurement principle and it's implementation and to illustrate the sensitivity and precision achievable with the CapArray meter some measurements are taken. Figure 6.1 shows the measurement setup.
The CapArray mentioned in Chapter 2 is fixed to a wooden board. It is connected to the CapArray meter by 24 cm long coaxial cables fixed to the array with conductive glue. The array is positioned so that the side connected to GND via the switch multiplexer is facing up and the side connected to the measurement port of the PCAP04 is facing down.
This is done to reduce the influence of parasitic capacitance caused by the human finger pressing the device.
The first row of the array seems to have been damaged internally while being used for the development of the CapArray meter and is excluded from any further testing.
The number 2 port on the right side of the CapArray meter was ripped of during previous experimentation. This has been patched by connecting the cable for the second column to port number 6 and changing the software to reflect this.
The CapArray meter is connected to a laptop PC running version 1.8.13 of the Arduino IDE under Linux.

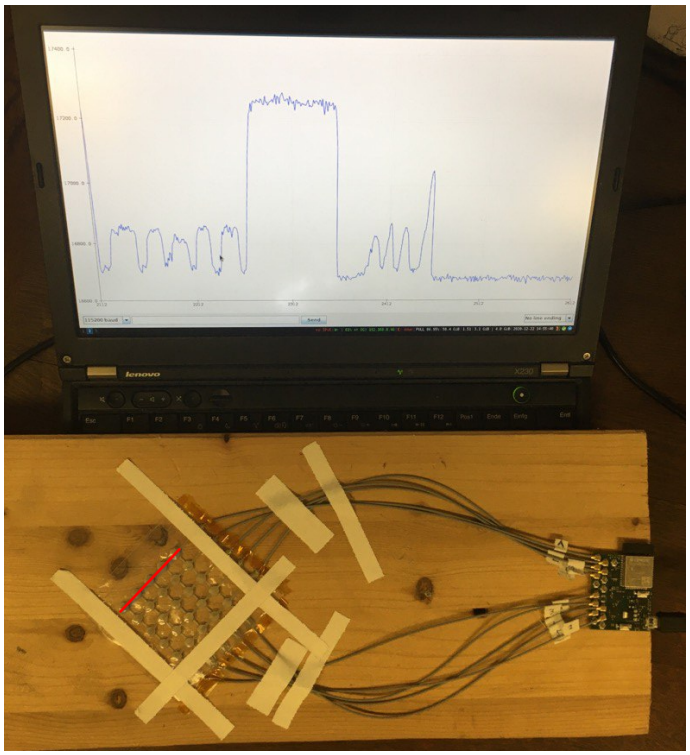

Figure 6.1: The measurement setup

6.1 Measurement of a Single Point

To illustrate the performance of the device only the point 3|3 is measured. For this row 3 of the CapArray connected to the left side port 3 is switched to GND. Column 3 of the array connected to the right side port 3 is switched to the measurement path.

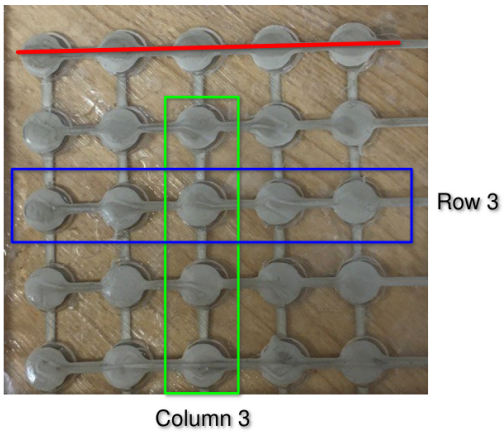

Figure 6.2: Illustration of row 3 and column 3, row 1 is crossed out due to being non functional

In the measurement shown in Figure 6.3 the points are lightly pressed with the tip of a finger for about 1 second each. The results are then plotted via the Arduino IDEs Serial Plotter. The time span shown in this graph is 25 seconds. First all points of column 3 are pressed then all points of row 3. Crosstalk between the points is mostly

caused by parasitic capacitance between the fingertip and the points of column 3. Even with this parasitic effect the pressure on the selected point 3|3 is clearly distinguishable.

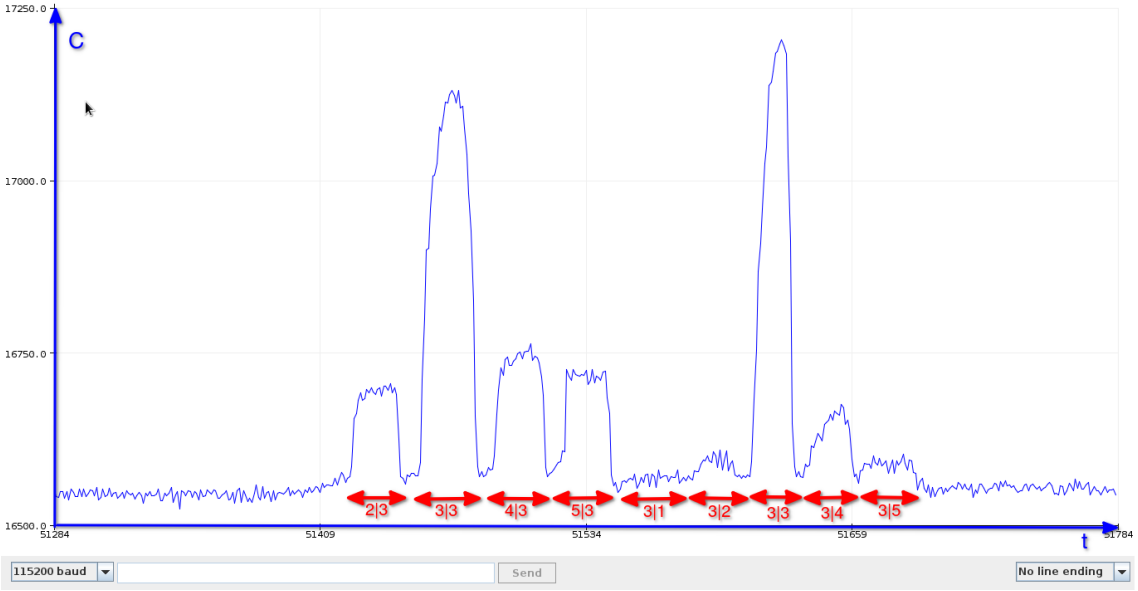

Figure 6.3: Crosstalk measurement of the point 3|3 using a finger to lightly press the points

The measurement shown in Figure 6.4 is similar. Instead of pressing with a finger the points are pressed with a flattened pencil that has the same diameter as the electrodes. Since the pencil is nonconductive no parasitic capacitance distorts the results. Figure 6.4 has been compressed along the Capacitance axis to match the scale of figure 6.3.
Crosstalk on the same column is reduced in this test but still noticeable. Crosstalk on the same row is almost nonexistent.
The overall deflection is also reduced compared to the previous measurement. The point 3|3 however remains clearly distinguishable.

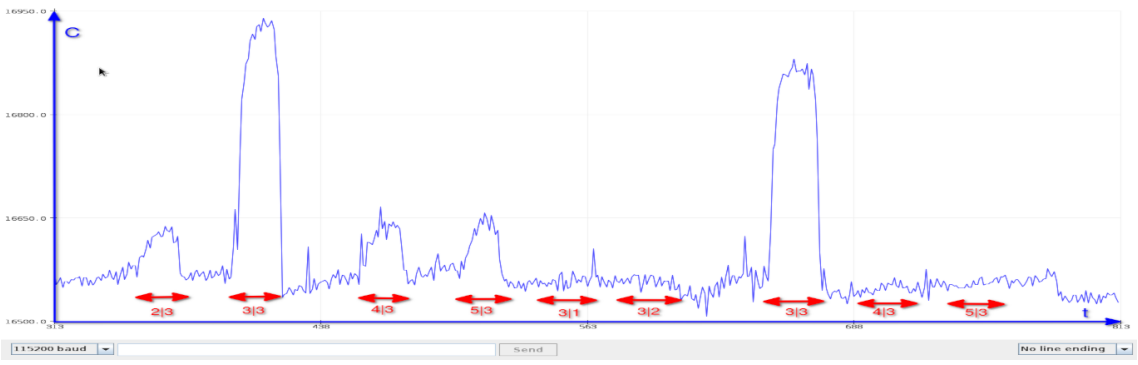

Figure 6.4: Crosstalk measurement of the point 3|3 using a flattened pencil to press the points

These graphs show that the measurement principle presented in Chapter 3 works well enough to distinguish single points. The crosstalk between the points directly connected to the measurement path (2|3; 3|3; 4|3 and 5|3 in these measurements) however is still significant. No explanation has been found for this effect.

7. Summary and Future Work

7.1 Summary

In this work the CapArray meter, a measurement device for soft and flexible 3D-printed pressure sensitive capacitor arrays was presented. A measurement principle was shown that can measure single points of such an array and effectively suppress crosstalk between the points. In Chapter 6 simple measurements were performed. These measurements show that the crosstalk compensation is functional, some crosstalk however still exists. They also show that the sensitivity and precision of the CapArray meter are adequate for finger touch sensing. Using this device testing of the CapArrays is no longer constrained to a laboratory. This device establishes a good platform for further development of the CapArray concept and provides the required flexibility for further enhancements.

7.2 Future Work

7.2.1 Accuracy

Achieving measurement accuracy has not been part of this work. In case future research requires accuracy characterisation of the devices parasitic capacitance as well as measurement drift could be done. This enables translating the measurement results from arbitrary units to the absolute capacitance value in Farad.

7.2.2 Further Development of the CapArray Meters Hardware

This section discusses possible hardware changes to the CapArray meter.

7.2.2.1 I2C

To circumvent the PCAP04 design flaw described in [8] the I2C bus should be replaced. The ESP-32 has two I2C interfaces and up to four for SPI. A combination of either I2C + I2C; I2C + SPI or only SPI could be used to replace the current bus topology. Alternatively an I2C address translator, such as the LTC4316, could be used to isolate the PCAP04 from the I2C bus.

7.2.2.2 Suppression of Stray Capacitance

With the current implementation the most significant stray capacitance, that of the coaxial wires, has already been suppressed. Board traces and components in the measuring path such as the CDC and the multiplexers still add a big capacitance offset. Reducing these by replacing the ground plane under the measurement section of the device by a V_{Guard} plane may speed measurement up significantly by reducing charge and discharge times.

7.2.2.3 Shielding

Outside error such as that introduced by moving a finger or metal object near the measurement path is detrimental to precision and accuracy. This outside influence can be reduced by shielding all sensitive parts of the device and the CapArray with a V_{Guard} layer.

7.2.2.4 Temperature Compensation

The PCAP04 includes a resistance to digital converter intended for resistive temperature sensing. For this either an on-chip resistor or an external precision resistor such as a PT1000 can be used. Temperature compensation software for the PCAP04s DSP is provided by the manufacturer. This could be used to implement compensation of the measurement drift caused by changes in temperature.

7.2.2.5 Battery Use and Low Power Optimization

The meter could be outfitted with a battery controller and battery for portable use. Optimising for low power may mean replacing the ESP32-WROOM module with a smaller MCU, as well as software changes to the measurement frontend. A battery powered circuit should include data storage for offline use or wireless communication, for example via Bluetooth Low Energy.

7.2.2.6 Measurement of Larger CapArrays

As mentioned in 4.2.6 expanding to 16x16 point CapArrays requires only more coaxial ports. Even further upgrades can be easily realised with more multiplexers.

7.2.3 Software Development

7.2.3.1 User Interface

The software written so far is fairly straightforward. A more sophisticated user interface could be written, supporting easy change of measurement parameters and better options for visualization and data export.

7.2.3.2 Wireless Communication

The hardware required for communicating via Wi-Fi, Bluetooth or Bluetooth Low Energy is already included on this version of the CapArray meter. Software could be written to use this option. Care should be taken that no communication occurs while the measurement is ongoing.

7.2.3.3 Changes to the Measurement Algorithm

The measurement algorithm could be changed in order to speed up scanning time and improve responsiveness of the device. Possibilities for this are:

- measuring the array at high speed (by taking fewer samples) and only performing a precise measurement when a change of capacitance has been detected
- measuring larger subsections of the array (by turning multiple switches to ON) and only measuring the points individually when a change of capacitance within the subsection has been detected

Both of these may also be used to reduce current consumption by putting the MCU and the PCAP04 into sleep mode between measurements while maintaining the same scanning time.

A. Software Documentation

```
#include <Wire.h>
#include "readoutroutine.h"

float tmp;

void setup() {
  Wire.begin();
  Serial.begin(115200);
  while (!Serial);
  Wire.beginTransmission(0x28);
  Wire.write(0xA3);//address information
  Wire.write(0xC0);//address information
  Wire.write(0x1D);//irrelevant
  Wire.write(0x01);//OX_RUN=1 is needed for guarding, OX runs permanently
  Wire.write(0x68);//48=180k discharge for PC0-PC4 68=30k ,no external resistor
  Wire.write(0x10);//charge resistor 10k
  Wire.write(0x00);//0x00 for grounded with external reference cap, PCAP04 compensation options disabled
  Wire.write(0x08);//CY_HFCLK_SEL=1 is needed for guarding 0x08 for 500kHz
  Wire.write(0x05);//0x05 for PC2+PC0 enabled for measurement
  Wire.write(0x62);//Number of measurements made (for more than 255 also the next Register) 0x62=98 measurements
  Wire.write(0x00);//also number of measurements
  Wire.write(0x00);//conv time, irrelevant in opcode triggered mode
  Wire.write(0x00);//conv time
  Wire.write(0x00);//conv time
  Wire.write(0x06);//Register 12  //0x06=12us discharge time
  Wire.write(0x18);//also part of discharge time, includes cdc trigger mode, 0x18=opcode triggered (by the opcode 0x8C)
  Wire.write(0x01);//Register 14  //0x01=6us precharge time
  Wire.write(0x08);//also part of precharge time //0x08=2fakes
  Wire.write(0x01);//0x01=6us fullcharge(plateau) time
  Wire.write(0x00);//also part of fullcharge time
  Wire.write(0xBF);//0xBF enables pulsed guard on each port
  Wire.write(0x00);//0x00 Guard Op-Amp Gain=1.00
  Wire.write(0x07);
```

```
34   Wire.write(0x01);
35   Wire.write(0x00);//0x00 RDC off
36   Wire.write(0x30);
37   Wire.write(0x73);
38   Wire.write(0x04);
39   Wire.write(0x50);
40   Wire.write(0x00);//DSP speed setting 0x00=fastest
41   Wire.write(0x5A);
42   Wire.write(0x00);
43   Wire.write(0x81);//0x81 DSP enabled by CDC
44   Wire.write(0x08);
45   Wire.write(0x08);
46   Wire.write(0x00);
47   Wire.write(0x57);//BG permanent enabled
48   Wire.write(0x40);
49   Wire.write(0x00);
50   Wire.write(0x00);
51   Wire.write(0x00);
52   Wire.write(0x71);
53   Wire.write(0x00);
54   Wire.write(0x00);
55   Wire.write(0x00);//0x00=asynchronus read disabled, if this is
      enabled result registers are only updated if empty
56   Wire.write(0x00);
57   Wire.write(0x00);
58   Wire.write(0x00);
59   Wire.write(0x00);
60   Wire.write(0x01);//0x01 Runbit is on, this turns on dsp and cdc
      frontend
61   Wire.write(0x00);
62   Wire.write(0x00);
63   Wire.write(0x00);
64   Wire.write(0x00);
65   Wire.endTransmission();
66   Wire.beginTransmission(0x28);
67   Wire.write(0x8C);
68   Wire.endTransmission();
69 }
70
71 void loop() {
72   float tmp = 0;
73   for (int i = 1; i <= 5; i++) {
74     for (int j = 1; j <= 5; j++) {
75       tmp = enablePoint(i, j);
76         switch (i) {
77           case 1:
78             Serial.print("A");
79             break;
80           case 2:
81             Serial.print("B");
82             break;
83           case 3:
84             Serial.print("C");
85             break;
86           case 4:
87             Serial.print("D");
88             break;
89           case 5:
90             Serial.print("E");
```

```
91              break;
92          }
93          switch (j) {
94            case 1:
95              Serial.print("A:");
96              break;
97            case 2:
98              Serial.print("B:");
99              break;
100           case 3:
101             Serial.print("C:");
102             break;
103           case 4:
104             Serial.print("D:");
105             break;
106           case 5:
107             Serial.print("E:");
108             break;
109         }
110         Serial.print(tmp);
111         Serial.print(",");
112     }
113   }
114   Serial.println();
115 }
```

Listing A.1: main.ino

```
1 float enablePoint(int left = 0, int right = 0)
2 {
3   int cdc = 2;
4   int NumberofResults = 3;
5   uint32_t Res[NumberofResults];
6   uint32_t Inv[NumberofResults];
7   float Result[NumberofResults];
8   uint32_t m1;
9   uint32_t l1;
10  uint32_t meas = 0;
11  float h1;
12  //Reset switches to known state (all lanes to guard)
13  Wire.beginTransmission(0x4D);
14  Wire.write(0x01);
15  Wire.write(0x00);
16  Wire.endTransmission();
17  Wire.beginTransmission(0x4D);
18  Wire.write(0x03);
19  Wire.write(0xFF);
20  Wire.endTransmission();
21  Wire.beginTransmission(0x4C);
22  Wire.write(0x00);
23  Wire.write(0x00);
24  Wire.endTransmission();
25  Wire.beginTransmission(0x4C);
26  Wire.write(0x02);
27  Wire.write(0xFF);
28  Wire.endTransmission();
29  delayMicroseconds(10);
30
```

```
31  switch (right) {//Write the appropriate Switch states for the
     right side ports
32    case 8:
33      Wire.beginTransmission(0x4D);
34      Wire.write(0x01);
35      Wire.write(0x01);
36      Wire.endTransmission();
37      Wire.beginTransmission(0x4D);
38      Wire.write(0x03);
39      Wire.write(0xFE);
40      Wire.endTransmission();
41      break;
42    case 7:
43      Wire.beginTransmission(0x4D);
44      Wire.write(0x01);
45      Wire.write(0x02);
46      Wire.endTransmission();
47      Wire.beginTransmission(0x4D);
48      Wire.write(0x03);
49      Wire.write(0xFD);
50      Wire.endTransmission();
51      break;
52    case 6:
53      Wire.beginTransmission(0x4D);
54      Wire.write(0x01);
55      Wire.write(0x04);
56      Wire.endTransmission();
57      Wire.beginTransmission(0x4D);
58      Wire.write(0x03);
59      Wire.write(0xFB);
60      Wire.endTransmission();
61      break;
62    case 5:
63      Wire.beginTransmission(0x4D);
64      Wire.write(0x01);
65      Wire.write(0x08);
66      Wire.endTransmission();
67      Wire.beginTransmission(0x4D);
68      Wire.write(0x03);
69      Wire.write(0xF7);
70      Wire.endTransmission();
71      break;
72    case 4:
73      Wire.beginTransmission(0x4D);
74      Wire.write(0x01);
75      Wire.write(0x10);
76      Wire.endTransmission();
77      Wire.beginTransmission(0x4D);
78      Wire.write(0x03);
79      Wire.write(0xEF);
80      Wire.endTransmission();
81      break;
82    case 3:
83      //uses alternative way of writing to MAX14661 to avoid
     sending 0x20 which crashes the PCAP04
84      Wire.beginTransmission(0x4D);
85      Wire.write(0x01);
86      Wire.write(0x00);//switch all to Guard
87      Wire.endTransmission();
```

```
88        Wire.beginTransmission(0x4D);
89        Wire.write(0x14);
90        Wire.write(0x0D);//switch only AB14 to PC2
91        Wire.write(0x00);
92        Wire.endTransmission();
93        Wire.beginTransmission(0x4D);
94        Wire.write(0x03);
95        Wire.write(0xDF);
96        Wire.endTransmission();
97        break;
98      case 2:
99        Wire.beginTransmission(0x4D);
100       Wire.write(0x01);
101       Wire.write(0x40);
102       Wire.endTransmission();
103       Wire.beginTransmission(0x4D);
104       Wire.write(0x03);
105       Wire.write(0xBF);
106       Wire.endTransmission();
107       break;
108     case 1:
109       Wire.beginTransmission(0x4D);
110       Wire.write(0x01);
111       Wire.write(0x80);
112       Wire.endTransmission();
113       Wire.beginTransmission(0x4D);
114       Wire.write(0x03);
115       Wire.write(0x7F);
116       Wire.endTransmission();
117       break;
118   }
119
120   switch (left) {//Write the appropriate Switch states for the left
        side ports
121     case 1:
122       Wire.beginTransmission(0x4C);
123       Wire.write(0x00);
124       Wire.write(0x01);
125       Wire.endTransmission();
126       Wire.beginTransmission(0x4C);
127       Wire.write(0x02);
128       Wire.write(0xFE);
129       Wire.endTransmission();
130       break;
131     case 2:
132       Wire.beginTransmission(0x4C);
133       Wire.write(0x00);
134       Wire.write(0x02);
135       Wire.endTransmission();
136       Wire.beginTransmission(0x4C);
137       Wire.write(0x02);
138       Wire.write(0xFD);
139       Wire.endTransmission();
140       break;
141     case 3:
142       Wire.beginTransmission(0x4C);
143       Wire.write(0x00);
144       Wire.write(0x04);
145       Wire.endTransmission();
```

```
146       Wire.beginTransmission(0x4C);
147       Wire.write(0x02);
148       Wire.write(0xFB);
149       Wire.endTransmission();
150       break;
151     case 4:
152       Wire.beginTransmission(0x4C);
153       Wire.write(0x00);
154       Wire.write(0x08);
155       Wire.endTransmission();
156       Wire.beginTransmission(0x4C);
157       Wire.write(0x02);
158       Wire.write(0xF7);
159       Wire.endTransmission();
160       break;
161     case 5:
162       Wire.beginTransmission(0x4C);
163       Wire.write(0x00);
164       Wire.write(0x10);
165       Wire.endTransmission();
166       Wire.beginTransmission(0x4C);
167       Wire.write(0x02);
168       Wire.write(0xEF);
169       Wire.endTransmission();
170       break;
171     case 6:
172       //uses alternative way of writing to MAX14661 to avoid
      sending 0x20 which crashes the PCAP04
173       Wire.beginTransmission(0x4C);
174       Wire.write(0x00);
175       Wire.write(0x00);//set all switches to Guard
176       Wire.endTransmission();
177       Wire.beginTransmission(0x4C);
178       Wire.write(0x14);
179       Wire.write(0x05);//switch only AB6 to GND
180       Wire.write(0x00);
181       Wire.endTransmission();
182       Wire.beginTransmission(0x4C);
183       Wire.write(0x02);
184       Wire.write(0xDF);
185       Wire.endTransmission();
186       break;
187     case 7:
188       Wire.beginTransmission(0x4C);
189       Wire.write(0x00);
190       Wire.write(0x40);
191       Wire.endTransmission();
192       Wire.beginTransmission(0x4C);
193       Wire.write(0x02);
194       Wire.write(0xBF);
195       Wire.endTransmission();
196       break;
197     case 8:
198       Wire.beginTransmission(0x4C);
199       Wire.write(0x00);
200       Wire.write(0x80);
201       Wire.endTransmission();
202       Wire.beginTransmission(0x4C);
203       Wire.write(0x02);
```

```
204       Wire.write(0x7F);
205       Wire.endTransmission();
206       break;
207   }
208
209   Wire.beginTransmission(0x28);
210   Wire.write(0x8C);
211   Wire.endTransmission();
212   delayMicroseconds(5600);//(12+6+6+1)*2*NoOfsamples+600(for I2C to
        send)
213   Wire.beginTransmission(0x28);
214   Wire.write(0x40);
215   Wire.endTransmission();
216   Wire.requestFrom(0x28, 4 * NumberofResults);
217   while (Wire.available()) {
218     for (int j = 0; j < NumberofResults; j++) {
219       for (int i = 0; i < 4; i++) {
220         meas = Wire.read();
221         meas = meas << (8 * i);
222         Res[j] = Res[j] + meas;
223       }
224     }
225   }
226   for (int j = 0; j < NumberofResults; j++) {
227     m1 = Res[j] >> 27;
228     l1 = Res[j] << 5;
229     l1 = l1 >> 5;
230     h1 = float(l1) / 0x8000000;
231     Result[j] = 10000 * (m1 + (h1));
232     Res[j] = 0;
233     if (j == cdc) {
234       return Result[cdc];
235     }
236   }
237 }
```

Listing A.2: readoutroutine.h

B. Schematic and Bill of Materials

Reference	Quantity	Value	Footprint
C1 C2 C4 C6 C10 C11 C12	7	0.1u	C_0402_1005Metric
C5	1	22u	C_0402_1005Metric
C3 C8	2	10u	C_0402_1005Metric
C7 C9	2	4.7u	C_0402_1005Metric
C_ref1	1	C_Small	C_0402_1005Metric
J1	1	USB_B_Micro	ZX62R-B-5P_30:HRS_ZX62R-B-5P(30)
J2 J4 J5 J6 J7 J8 J9 J10 J11 J12 J13 J14 J15 J16 J17 J18 J19	17	Conn_Coaxial	Connector_Coaxial:U.FL_Molex_MCRF_73412-0110_Vertical
Q1 Q2	2	BC817	Package_TO_SOT_SMD:SOT-23
R1 R8 R9	3	10k	R_0402_1005Metric
R7 R10 R11	3	0	R_0402_1005Metric
R2	1	22k1	R_0402_1005Metric
R3	1	47k5	R_0402_1005Metric
R4	1	1k	R_0402_1005Metric
R5 R6	2	4k7	R_0402_1005Metric
SW1 SW2	2	SW_Push	PTS636SM25SMTRLFS
U1	1	LT1117-3.3	Package_TO_SOT_SMD:SOT-223
U2	1	USBLC6-2SC6	Package_TO_SOT_SMD:SOT-23-6
U3	1	ESP32-WROOM-32	RF_Module:ESP32-WROOM-32
U4	1	CP2102N-A01-GQFN24	Package_DFN_QFN:QFN-24-1EP_4x4mm_P0.5mm_EP2.6x2.6mm
U5	1	PCAP04	Package_DFN_QFN:QFN-24-1EP_4x4mm_P0.5mm_EP2.6x2.6mm
U6 U7	2	MAX14661ETI+	MAX14661ETI_:QFN40P400X400X80-29N

Figure B.1: Bill of Materials for the second revision of the CapArray Meter

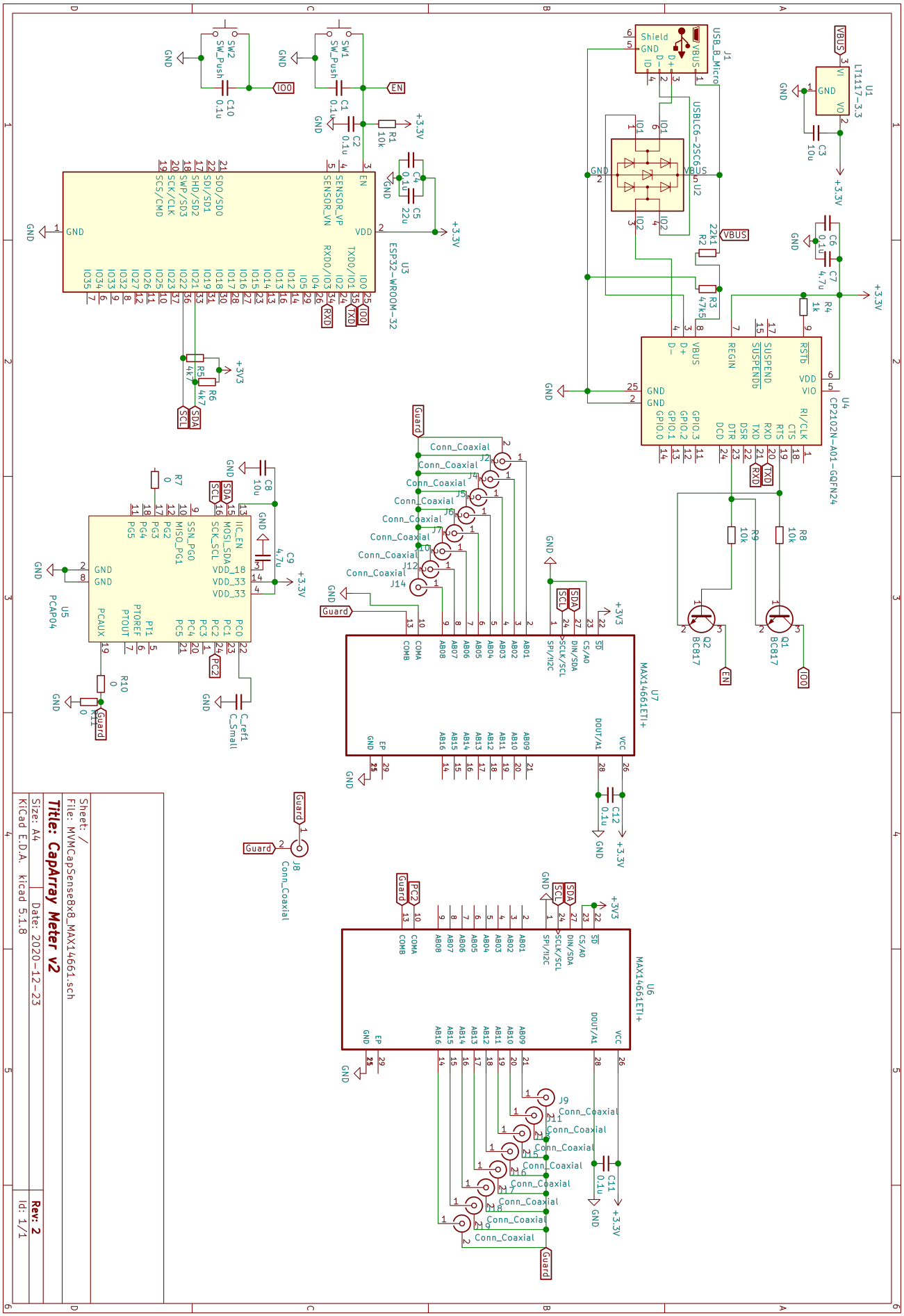

Figure B.2: Full schematic of the second revision of the CapArray Meter

Bibliography

[1] Hongye Sun, Zongyou Han, and Norbert Willenbacher. "Ultrastretchable Conductive Elastomers with a Low Percolation Threshold for Printed Soft Electronics". In: *ACS Applied Materials & Interfaces* 11.41 (2019). PMID: 31566949, pp. 38092–38102. eprint: https://doi.org/10.1021/acsami.9b11071.

[2] Kevin Bull. *Methods of accurately measuring capacitive RH sensors*. 2006.

[3] *PCap04 Capacitance-to-Digital Converter*. PCAP04. Rev. 1-03. ams. 2018.

[4] Wilfried Plaßmann and Detlef Schulz. *Handbuch Elektrotechnik*. Springer, 2016.

[5] https://commons.wikimedia.org/wiki/File:Series_RC_resistor_voltage.svg.

[6] E. Pritchard et al. "Flexible capacitive sensors for high resolution pressure measurement". In: *SENSORS, 2008 IEEE*. 2008, pp. 1484–1487.

[7] http://www.wellshow.com/spec/cable/D1320WS5BS.pdf. Last accessed: 2020-12-05.

[8] *PCAP04 Errata Document of Datasheet Version 3-00 or later*. PCAP04. Rev. 1-01. ams. 2019.

[9] *ESP32-WROOM-32 Datasheet*. ESP32-WROOM-32. Rev. 3.0. Espressif Systems. 2020.

[10] *USBXpress Family CP2102N Data Sheet*. CP2102N. Rev. 1.5. Silicon Laboratories. 2020.

[11] *MAX14661 Beyond-the-Rails 16:2 Multiplexer*. MAX14661. Rev. 2. Maxim Integrated. 2015.

[12] usb.org.

[13] *800mA Low Dropout Positive Regulators Adjustable and Fixed 2.85V, 3.3V, 5V*. LT1117-3.3. Rev. D. Linear Technology. 1993.

[14] *USBLC6-2 Datasheet*. USBLC-6. Rev. 6. STMicroelectronics. 2020.