

PADME

Agnes Gårdebäck
Gustav Dänzel
Jacob Stuart

Nacka Gymnasium

16 mars 2012

Innehåll

1	Prolog	2
2	Abstrakt	2
3	Myoner	2
4	Particle Flux Demonstrator	3
5	Uppgift	4
6	Konstruktion	6
6.1	Gränssnittet	6
6.2	Programmering	7
6.3	Elektronik	8
6.4	CAD och 3D-Utskrift	10
7	Slutprodukt	11
A	Programkod PADME	12

Figurer

1	Particle Flux Demonstrator med kontakt för inkoppling av PADME	5
2	Flödeschema över användning av PADME	6
3	Rendering av Teensy och skärm monterade på kopplingsplatta	8
4	Kopplingschema över PADME	9
5	Demoplatta respektive lödning vid produktionen av PADME prototyp	10
6	Utskrift av skalet till PADME i 3D-skrivare	11
7	CAD-ritning och rendering av PADME i Autodesk Inventor .	11
8	Första fungerande prototypen av PADME	12

1 Prolog

Det var en gång i ett land inte alls långt långt borta, tre figurer vid namn Agnes, Gustav och Jacob. När vi först träffar våra hjältar har de precis börjat delta i informationsmöten som ska mynna ut i ett projektarbete om komisk strålning, föga anar de dock svårigheterna de förväntas övervinna för att utföra de mätningar som krävs för denna uppgift. Ingen hade tidigare varnat dem för frustrationen och tristessen som skulle överväldiga dem när de manuellt skrev ner varje träff som detektorn de fick låna registrerade. De visste ännu inte om hur lätt en liten harmlös lysdiod kunde övertyga en om att dess ända syfte var att driva en till vansinne, bara genom att blinka i oregelbundna intervaller. Som tur är så fann sig inte dessa hjältar i en sådan behandling och förnedring. Efter upprörda rådslag enades de om att en gång för alla befria mänskligheten från denna pina och de axlade ansvaret att utveckla ett alternativ till den tålmodsprövning som detektorerna hittills försåg sina offer med. Det underverk som byggdes befriade människorna från fortsatt lidande och beskrivs i följande rapport.

2 Abstrakt

PADME (PArTicle Demonstrator Monitoring Extension) projektet har som syfte att utveckla en tilläggsenhet till de portabla partikeldetektorer (PFD enheter) som används i utbildningssyfte av Vetenskapens Hus. Tilläggsenhetens syfte är att ersätta detektorernas användargränssnitt vilket innefattar fyra lysdioder som blinkar vid varje registrerad partikel. Det nya gränssnittet bygger istället på en digital display med integrerad timer eftersom detta förenklar användningen av detektorerna samt möjliggör mätningar med högre precision. Övriga funktioner inkluderar en USB-port för att direkt kunna överföra mätdata till en dator samt en nätkontakt för att kunna driva detektorn från en extern strömkälla under längre mätningar.

PADME:n har konstruerats av elever från Nacka Gymnasium som ett projektarbete i fysik och teknik med material från Vetenskapens Hus.

3 Myoner

De partiklar där man inte funnit att de är uppbyggda av mindre beståndsdelar är elementära vilket betyder osammansatt eller grundläggande och partiklarna kallas därför elementarpartiklar. Idag har man funnit många av dessa partiklar och det finns indikationer på att det finns många fler vars existens man ännu inte kunnat bevisa.

Elementarpartiklarna delas upp efter deras spinn: en kvantfysikalisk egenskap som anger en form av partikelns inre rörelsemängdsmoment. De som har halvtaligt spinn kallas för fermioner och de med heltaligt spinn kallas

för bosoner. Fermionerna delas upp i ytterligare två undergrupper vid namn kvarkar och leptoner. Leptonerna skiljer sig främst från kvarkarna genom att de endast växelverkar svagt medan kvarkarna även kan växelvärka starkt och på så sätt bilda sammansatta partiklar. De sammansatta partiklarna kallas hadroner och delas upp i baryoner som består av tre olika kvarkar respektive mesoner vilka består av en kvark och en antikvark. Alla elementarpartiklar har en motsvarande antipartikel med motsatt laddning.

Leptonerna är en grupp med tre familjer av elementarpartiklar där varje familj består av en laddad partikel med motsvarande neutrino och deras antipartiklar. Elektronens antipartikel har ett eget namn och kallas för positron. En neutrino är en oladdad partikel med mycket liten massa och eftersom de är neutrala växelverkar de endast svagt med annan materia vilket gör neutriner svåra att upptäcka. De tre laddade leptonerna kallas för elektron, myon och tau-lepton.

De partiklar som mäts med PFD:n är myoner som likt elektronen är negativt laddade men har en massa ca 200 gånger större än elektronens. De bildas vid kosmisk strålning då protoner med mycket stor rörelseenergi träffar atomkärnor i jordens atmosfär vilket ger upphov till ett sönderfall som sänder en partikelskur ner mot jordens yta. Partikelskuren består av många olika partiklar till exempel myoner. Myonerna har en mycket kort livslängd och skulle inte nå ner till ytan om det inte vore för Einsteins speciella relativitetsteori vilket gör att de mäter tiden i ett annat referenssystem. Då myonerna färdas relativt nära ljusets hastighet så uppfattar de på grund av längdkontraktionen och tidsdilatationen, avståndet till jorden som mycket kortare än vad vi på jorden gör. En del myoner når därför ner och kan detekteras vid jordens yta. De andra partiklarna som bildas i sönderfallet har oftast ännu kortare livslängd än myonerna och når därför sällan ner till ytan vilket gör att man mäter detektioner av myoner när man undersöker kosmisk strålning på marknivå.

4 Particle Flux Demonstrator

Arbetet baseras på den portabla partikeldetektorn PFD (Particle Flux Demonstrator) vilken utvecklades av KTH som en del av SEASA projektet vilket bland annat innefattade att skicka med en partikeldetektor upp i rymden tillsammans med Christer Fuglesang. Detta gjordes under hans resa till ISS år 2006. Detektorns syfte var att demonstrera skillnader i strålningsintensitet i olika delar av omloppsbanan samt inom ISS. Skol elever kunde sedan jämföra mätningarna med egna mätningar utförda med identiska detektorer.

Detektorerna har sedan projektets slut utnyttjats av Vetenskapens Hus för att ge elever möjlighet att göra egna mätningar av partikelstrålning i olika syften och arbeten. Vetenskapens hus är en utbildningsverksamhet som drivs

genom ett samarbete mellan Stockholms Universitet och Kungliga Tekniska Högskolan.

Detektorerna är en typ av scintillationsräknare vilka reagerar på de ljusblixtar som bildas när joniserande högenergetiska partiklar passerar ett fluorescerande material. De fotoner som då emitteras har en våglängd på 425 nm och leds genom ett fotomultiplikator-rör (PMT) vilket förstärker och omvandlar dem till en mätbar elektrisk puls vars styrka är proportionell mot den ursprungliga pulsen. I PFD-enheterna används en behandlad plast som fluorescerar när negativt laddade partiklar passerar genom den eftersom de partiklar man vill undersöka är myoner vilka har en negativ elementarladdning. För att filtrera ut de högenergetiska partiklarna från kosmisk strålning ur den övriga bakgrundstrålningen av betapartiklar och dylikt används två separata scintillatorplattor. Detta gör att bara myonerna som avses undersökas har tillräckligt med energi för att ta sig igenom båda plattorna vilket gör att det bildas två ljusblixtar i tät följd så att träffen registreras. Praktiskt görs detta genom att kondensatorer kopplas till fotomultiplikatorrören vilka förlänger de elektriska pulserna så att de överlappar varandra. Den puls som då bildas är tillräckligt stark för att nå över tröskelvärdet som finns inbyggt i detektorn och pulser från enbart en platta eller partiklar med låg energi når inte över detta tröskelvärde så att impulsen ignoreras. Uppbyggnaden med två parallella scintillatorplattor medför även att enbart partiklar som passerar detektorn inom ett visst vinkelintervall registreras.

Av byråkratiska skäl kunde detektorn inte innehålla ett automatiserat räkneverk för att få tillstånd att transporteras till ISS och därför redovisas istället detekterade partiklar med ett system av lysdioder på ovansidan och räknas manuellt. När detektorn registrerar en passerande partikel visas detta med en blinkning från en av lysdioderna och förutom denna lysdiod finns ytterligare tre lysdioder som blinkar vid var tionde, hundra samt tusende registrerade partikel. Detta system gör längre mätningar mycket svåra att genomföra eftersom konstant övervakning krävs, det introducerar också en möjlig felkälla beroende på användarens koncentration förmåga.

5 Uppgift

Då vi ville utföra betydligt längre mätningar när vi kontaktade Vetenskapens Hus insåg vi att sådana inte var praktiskt genomförbara. Vi uttryckte då ett intresse för att utveckla ett digitalt räkneverk till den PFD som vi erbjöds låna för mätningar. Det visade sig då att det redan fanns ett intresse för detta och att idén redan hade godkänts men inte tagit sig längre än till planeringsstadiet. Ett samarbete med Vetenskapens Hus inleddes därför för att utveckla dessa räkneverk.

De kriterier Vetenskapens Hus ställde på arbetet var att tillsatsen skulle vara löstagbar och att detektorn skulle fungera som tidigare utan tillsat-



Figur 1: Particle Flux Demonstrator med kontakt för inkoppling av PADME

sen. Detta gjorde att det inte var möjligt att göra omfattande ändringar i detektorn utan endast avläsa de signaler som redan fanns.

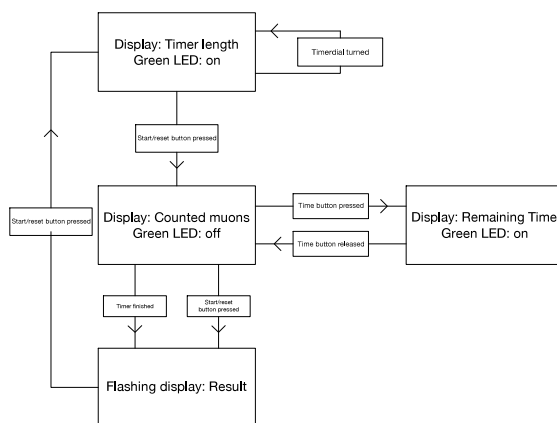
Det vi ville uppnå med tillsatsen var att förenkla utförandet av mätningar med PFD:n genom att visa antalet registrerade partiklar med en display istället för det optiska räkneverk som fanns. Detta skulle också möjliggöra längre mätningar samt ta bort den eventuella felkällan associerad med manuellt räknande. För att göra mätningar med maximal noggrannhet behövdes också en funktion för att göra precist tidsbegränsade mätningar samt ett gränssnitt för att bestämma deras längd. En ytterligare efterfrågad funktion i tillsatsen var att i realtid kunna överföra mätdata till en dator. För att kunna genomföra detta byggdes tillsatsen runt en programmerbar mikroprocessor. Skalet som skulle husera mikroprocessorn samt övrig elektronik såsom knappar och display designades i ett CAD-program för att sedan skrivas ut i plast med en 3D-skrivare. Vid formgivningen av tillsatsen hölls i åtanke att få tillsatsen så liten som möjligt för att kunna passa i det begränsade utrymmet på detektorns ovansida.

6 Konstruktion

6.1 Gränssnittet

För att hantera enhetens timer-funktion skapades ett gränssnitt baserat på två tryckknappar och en vridknapp. När enheten startas visas en vald tidslängd på displayen, tidslängden kan ställas till flertalet förinställda värden genom att vrida på vredet till höger om displayen. Mätningen startas sedan med start/reset-knappen och displayen övergår då till att visa antalet detekterade träffar tills timern avbryter mätningen och resultatet då blinkar på displayen för att visa att mätningen är slutförd. För att påbörja en ny mätning trycker användaren åter på start/reset-knappen och en ny tidslängd kan väljas. Man kan även göra mätningar utan tidsbegränsning genom att vid inställning av timern ställa denna till noll, mätningen stoppas då med start/reset-knappen vilken även kan användas för att avbryta tidsbegränsade mätningar i förväg. Efter att en mätning avbrutits manuellt med start/reset-knappen visas resultatet blinkande på displayen tills användaren väljer att påbörja en ny mätning på samma sätt som efter en tidsbegränsad mätning.

Det är också möjligt att se kvarvarande tid av en tidsbegränsad mätning genom att trycka på timer-knappen då antalet återstående sekunder visas på displayen så länge knappen är nedtryckt. När displayen visar tid av något slag tänds en grön lysdiod för att detta ska kunna skiljas från antal räknade partiklar.



Figur 2: Flödeschema över användning av PADME

Utöver de två knapparna, vredet och displayen finns på tillsattens framsida också en röd lysdiod för att indikera låg batterinivå, en miniUSB-port samt en nätkontakt för extern strömförsörjning.

6.2 Programmering

Som grund för tillsatsen användes en mikroprocessor förmonterad på ett mindre kretskort med miniUSB-port och kontaktbleck för koppling till annan elektronik. Dessa kontaktbleck kan ställas till anoder eller katoder samt avläsas för information om passerande spänning med hjälp av mikroprocessorn vilken programmerades i språket Arduino. Språket är en modifierad variant av C framtagen för Arduino utvecklings-kort vilka liknar de kort av typen Teensy vi använde.

I början utfördes programmeringen med hjälp av en demoplatta på vilken alla komponenter kunde kopplas temporärt utan lödning. På det sätt testades styrningen av displayen med processorn samt att få processorn att agera utifrån knapptryckningar och olika värden från potentiometern. På demoplattan testades också olika kombinationer av knappar och vred för att komma fram till ett användarvänligt och intuitivt gränssnitt.

Eftersom den sju-segments display som användes saknade elektronik för att ta emot information seriellt kopplades den direkt till Teensyn med en anod per siffra och en gemensam katod per segment. För att tända en av de fyra siffrorna kopplas denna till ström och segmenten som ska lysa jordas, detta innebär att alla siffror inte kan lysa samtidigt eftersom de olika siffrorna delar kopplingar för att jorda segmenten. Programmet blinkar därför de fyra siffrorna individuellt i snabb följd vilket skapar en illusion av att hela displayen lyser. För att kunna göra detta måste programmet först dela upp de tal som ska visas i dess bestående siffror vilket görs med hjälp av restdivision.

Efter att talet delats upp i siffror går programmet igenom siffrorna och jordar rätt segment utifrån en tabell över vilka segmentkombinationer som bildar en siffra. När rätt segment är jordade tänds siffran i tre millisekunder innan den släcks och nästa siffras segment jordas. Om siffrorna blinkar med kortare pulser gör detta att displayen upplevs ljussvag medan längre pulser kan få siffrorna att flimra.

Mellan varje tändning av en siffra kontrollerar programmet flertalet gånger om statusen på någon knapp eller vred har förändrats och om en annan del av programmet därför ska köras. För att få ett mycket noggrant värde på när detektorn registrerar en partikel används en funktion i processorn kallad en interrupt. En interrupt är en funktion vilken låter viss kod köras som en subrutin direkt när en förändring av spänningen på en specificerad koppling till mikroprocessorn sker. Denna subrutin avbryter tillfälligt övrig kod och kan därför användas för att få exakta tidsvärden på när något inträffade.

Kod: Se bilaga A

6.3 Elektronik

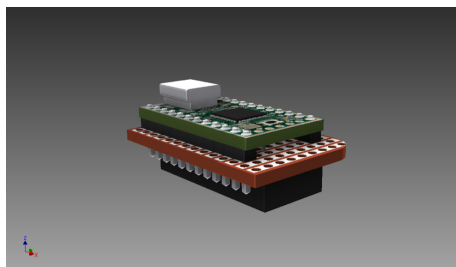
Microprocessorn har 23 kontaktbleck för input och output samt en miniUSB-kontakt vilken gör att den kan kommunicera med en dator. I PADME:n så drivs den från detektorns batteri men den kan även drivas av USB-porten. Eftersom mikroprocessorn klarar max 5V så används en spänningsomvandlare mellan processorn och batteriet vilket sänker den inkommande spänningen till 5V. Då USB-porten normalt bidrar med 5V till processorn när den är kopplad till datorn skulle spänning från båda källorna samtidigt skada processorn. Detta löses genom att man tar bort den positiva ledaren från den förlängning som går mellan processorns USB-port och USB-porten på PADME:ns utsida. Förlängningen består av en sladd med mini-USB hona och hane och har alltså lödts med bara tre ledare istället för normala fyra; två för utbyte av information och en för jord.

Displayen kopplades till mikroprocessorn genom att båda monterades på motsatta sidor av en kopplingsplatta vilken är ledande i en dimension via en metallyta som kan skrapas bort för att begränsa utsträckningen av kontakten. Pinnar löddes fast på mikroprocessorns kopplingsytor för att underlätta lödningen. Därefter programmerades processorn till att hantera displayen efter vilka pinnar som kom i kontakt med displayens stift. Mellan processorns och displayens anoder användes motstånd för att inte skada de spänningskänsliga segmenten.

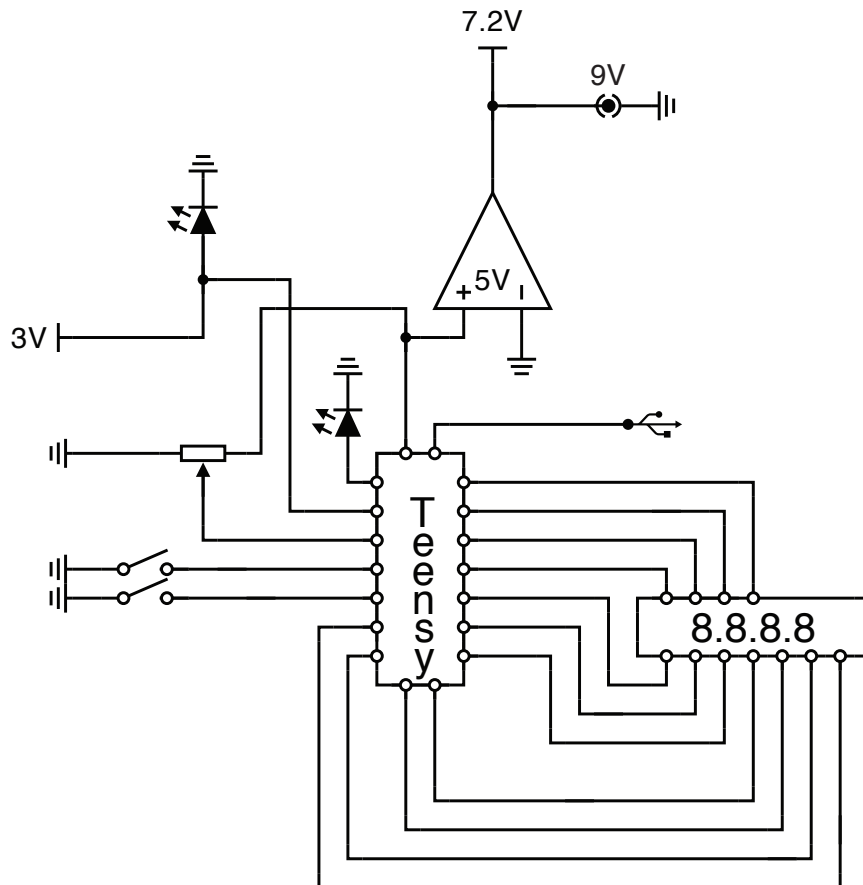
Knapparna kopplas till mikroprocessorn som pluspol samt jord och är byggda så att när man trycker ner dem kopplas de till jord vilket leder till ett spänningsfall vid mikroprocessorn som denna kan avläsa. Potentiometern använder till skillnad från knapparna inte mikroprocessorn som pluspol utan kopplas istället till spänningsomvandlarens anod. Strömmen leds sedan till jord genom en kolskena med högt motstånd och en kolsläde som dras över skenan när potentiometer vrids och avståndet till pluspolen respektive jord varierar då och gör att även spänningen vid släden varierar. Mikroprocessorn är kopplad till släden och fungerar då som en voltmeter vilken avläser spänningen, alltså kan mikroprocessorn avgöra hur potentiometern är vriden.

Den gröna lysdioden som indikerar när tid visas på displayen är kopplad till en av mikroprocessorns pinnar och drivs med denna. Den röda lysdioden som visar batterispänning är istället direkt kopplad till den ledare som går till motsvarande lysdiod på detektorn men som skyms av tillsatsen.

Från detektorn behövs information till PADME:n om antalet träffar, sta-



Figur 3: Rending av Teensy och skärm monterade på kopplingsplatta

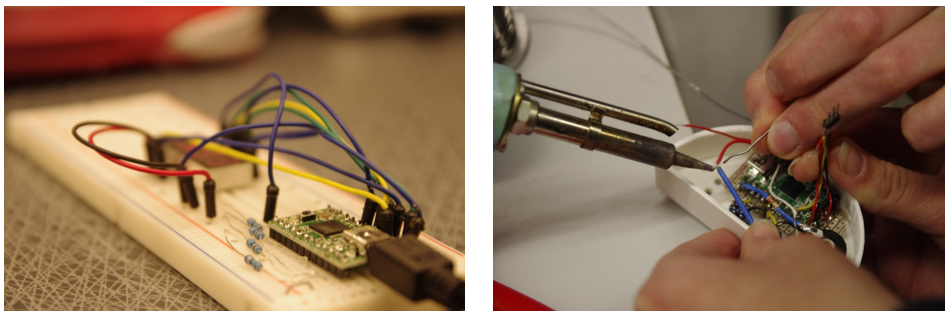


Figur 4: Kopplingschema över PADME

tus hos den röda lysdioden samt ström från batteriets. Därför har sladdar dragits från ledarna i detektorn till och från batteriet, ledaren som skickar impulser till lysdioden vilken blinkar vid varje träff samt ledaren som jordar den röda lysdioden vid låg spänning över batteriet. Dessa sladdar leds upp genom ett skruvhål i detektorns ovansida och slutar i en kontakt (kombinerad hona/hane för att undvika felkoppling vid anslutningen) med motsvarande kontakt hos PADME:n.

Nätadaptern som kopplas till nätkontakten på tillsatsen för att ansluta en extern strömkälla ger 9V likström och gör det möjligt att driva PADME från ett vanligt vägguttag. Nätkontakten kopplas in på motsatt sida om spännings-omvandlaren sett från mikroprocessorn eftersom den som tidigare nämnts bara tål 5V. Då batteriet är laddningsbart så har det en spänning på 7,2V d.v.s. en spänning lägre än 9V vilket gör att det riskerar att överladdas när den externa strömkällan kopplas in så att en vätgasutveckling startas

och utgör en explosionsrisk. För att förhindra att en sådan olycka inträffar kopplas därför en diod in vid batteriets anod. En diod släpper inte igenom någon ström om spänningen är högre vid diodens anod än katod, genom att koppla en diod mellan batteriet och övriga kretsen blockeras ström genom batteriet när den högre spänningen från närliggande kontakten är närvarande i kretsen. När ingen extern strömtillförsel finns är spänningen lägre vid batteriets katod än dess anod vilket gör att dioden återigen släpper igenom ström. Detta sker direkt när spänningen sjunker under 5 volt vilket betyder att den externa strömförsörjningen kan anslutas och fränkopplas löpande under drift utan att påverka detektorn eller PADME:n.



Figur 5: Demoplatta respektive lödning vid produktionen av PADME prototyp

6.4 CAD och 3D-Utskrift

För att skapa de 3D-ritningar som används för utskriften av skalet använde vi oss av CAD-programmet Autodesk Inventor. CAD står för Computer-aided Design och innefattar program som används för att göra digitala ritningar och modelleringar, Inventor är anpassat framför allt för utveckling av tekniska konstruktioner.

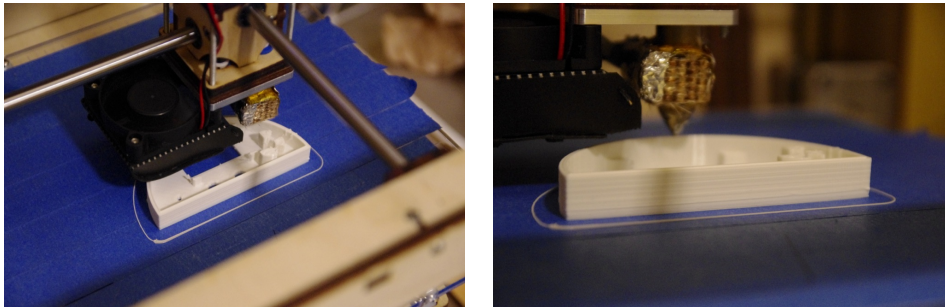
Genom att rita upp olika objekt kan man med programmet undersöka hur deras passform överensstämmer med varandra och därmed undvika onödiga prototyper eftersom kollisioner och liknande kan upptäckas innan produktion. Efter att med hjälp av noggranna mätningar modellerat mikroprocessorn, displayen, knapparna och potentiometern kunde på så sätt testas om uppbyggnaden av skalet fungerade i praktiken samt undersöka lämplig placering av komponenter.

För prototypen och produktion av skalet användes en 3D-skrivare vilken skapar ett fysiskt föremål på samma sätt som en vanlig laserskrivare skapar en bild. Istället för droppar av färg använder 3D-skrivaren smält plast som i strängar placeras i lager för att skapa ett föremål vilket också har höjd.

Skalet är uppdelat i två delar: en övre bestående av ovansida och väggar samt en undre bottenplatta. Denna design valdes eftersom 3D-skrivaren inte

kan skapa detaljer som saknar bärande stöd under utskriften samt att det förenklar montering och reparation. Den övre delen skrivs ut med "taket" nedåt och komponenterna monteras i den nedifrån och upp. Den övre delen är gjord för att passa precis innanför detektorns cirkulära kant men täcker bara strax över halva detektorns ovansida eftersom plats måste lämnas åt de reglage som används för att styra detektorn. Ovansidan är perforerad med öppningar så att de olika komponenterna når ut och på insidan finns stöd för justering av höjd av komponenter utanför tillsatsen samt för att göra monteringen mer stabil. Bottenplattan täcker inte hela tillsatsen för att lämna en öppning där kontakten med sladdar till detektorn kan kopplas in samt är försedd med hål för att ge plats åt lysdioderna som är placerade under tillsatsen.

Då det vid flera tillfällen under utvecklingen lades till nya funktioner som krävde anpassning av skalet så skrevs trots testerna i CAD-programmet flera prototyper ut. Noggrannheten hos mätningarna samt noggrannheten i utskrifterna från 3D-skrivaren gjorde även att marginalerna behövde justeras en del då öppningarna tenderade att bli för små, något som medförde omfattande slipningsarbete innan prototyperna kunde monteras.



Figur 6: Utskrift av skalet till PADME i 3D-skrivare



Figur 7: CAD-ritning och rendering av PADME i Autodesk Inventor

7 Slutprodukt

När vi först påbörjade arbetet med PADME var vår vision att utveckla en tillsatts som var användarvänlig och möjliggjorde långtidsmätningar. Under arbetets gång har målen förändrats och kommit att innefatta även andra funktioner så som timer, överföring av data via mini-USB och möjlighet till inkoppling av en extern strömkälla. Givetvis finns ytterligare fler funktioner som skulle kunna tillföras, till exempel funktioner för att lagra mätdata från flera tillfällen, detta var dock inte realistiskt med de tidsbegränsningar som rådde. Den slutprodukt vi uppnått möter väl de förväntningar vi hade när projektet påbörjades och ska förhoppningsvis även möta förväntningarna hos slutanvändarna.



Figur 8: Första fungerande prototypen av PADME

A Programkod PADME

PADME.ino

```
1 //initiating variables
2 const int displayPins [] = {20, 18, 9, 6, 13, 15, 16}; //list
   of pins to ground to light SSD segments A-G
3 const int powerPins [] = {11, 14, 17, 0}; //list
   of pins to power to light digits 1-4
4 const int interruptPin = 3; //pin
   connected to pushbutton (must be interrupt enabled pin)
5 const int resetPin = 1;
6 const int timeshowPin = 5;
7 const int potentiometerPin = 19;
8 const int ledPin = 21;
9
10 int tal [2][4] = {
11     {10, 10, 10, 0},
12     {10, 10, 10, 0},
13     }; //empty array for storing individual
   digits from value x, starts filled with a single digit
   zero
14
15 const int siffror [11][7] = { //two-dimensional array
   containing lists of digital write states (1/0) for creating
   digits
16     {0,0,0,0,0,0,1}, //0
17     {1,0,0,1,1,1,1}, //1
18     {0,0,1,0,0,1,0}, //2
19     {0,0,0,0,1,1,0}, //3
20     {1,0,0,1,1,0,0}, //4
21     {0,1,0,0,1,0,0}, //5
22     {0,1,0,0,0,0,0}, //6
23     {0,0,0,1,1,1,1}, //7
24     {0,0,0,0,0,0,0}, //8
25     {0,0,0,0,1,0,0}, //9
26     {1,1,1,1,1,1,1}, //empty (digit off)
27     };
28
29
30 volatile unsigned long currentTime; //variable for storing
   relative current time
31 volatile unsigned long nextTime;
32 unsigned long buttonPressed;
33 long timerLength;
34
35 volatile unsigned int counted = 0; //variable that stores
   current counted value
36 boolean timerAvilable = true;
37
38 unsigned long onTime = 0; //variable storing last time
   a digit was lit
```

```

39 int numberSSD = 4; //variable storing which
    display digit to light
40
41 int potentiometer = 0; //variable storing value
    read from potentiometer
42
43 //initiating functions
44
45 void valueSplit(int z, int x) { //function that split values
    into composing numbers
46
47     tal[z][0] = (x - x%1000)/1000; //stores first
        digit of value x in array "tal"
48     tal[z][1] = (x%1000 - x%100)/100; //stores second
        digit of value x in array "tal"
49     tal[z][2] = (x%100 - x%10)/10; //stores third
        digit of value x in array "tal"
50     tal[z][3] = (x%10); //stores forth
        digit of value x in array "tal"
51
52     for (int i = 0; i <= 2; i++) {
53
54         if(tal[z][i] == 0) { //loops over
            digits in array tal and replaces first three 0's
            before the actual value with a blank digit
55             tal[z][i] = 10; //digit 10 is
                all segments off digit
56         } else {
57             break;
58         }
59     }
60 }
61
62 void screenUpdate(int z) { //function that
    moves lit digit one position left
63
64     if (currentTime - onTime > 2) { //checks if
        digit has been lit for more then 1 ms
65
66         digitalWrite(powerPins[numberSSD - 1], 0); //Turns off
            previously lit digit
67
68         if(numberSSD >= 4) { //resets
            variable for digit to light if all digits on display
            has been lit
69             numberSSD = 0;
70         }
71
72         for (int i = 0; i <= 6; i++) { //loops over
            display segments A-G using variable i and checks if
            segment should be on or off
73             digitalWrite(displayPins[i], siffror[tal[z][
                numberSSD]][i]); //sets pin for segment i on
                or off

```

```

74     }
75
76     digitalWrite(powerPins[numberSSD], 1); //turns on digit
       from numberSSD
77
78     onTime = currentTime; //stores time
       digit was lit in variable "ontime"
79     numberSSD++;
80 }
81 }
82
83 void trigger() {
84     if (currentTime > nextTime) { //checks if 3 ms
       has passed since last trigger
85         counted++; //increases
           counted variable with +1
86         nextTime = currentTime + 3; //set
87         Serial.print(counted); //send current
           counted button-presses to connected computer
88         Serial.print(" "); //send space
           character to computer to help python script
89         Serial.println(currentTime); //send time
           since boot (ART) to connected computer
90         valueSplit(0, counted % 2000); //divides
           counted value into composing digits and saves to
           array "tal" for use by function displayUpdate
91     }
92 }
93 //startup-routine
94 void setup() {
95
96     Serial.begin(2400); //open serial port
       for communication with connected computer, sets data rate
       to 38400 bps
97     pinMode(interruptPin, INPUT); //set pins for
       output/input
98     pinMode(resetPin, INPUT_PULLUP);
99     pinMode(timeshowPin, INPUT_PULLUP);
100    pinMode(potentiometerPin, INPUT);
101    pinMode(ledPin, OUTPUT);
102
103    for (int i = 0; i <= 6; i++) { //loops over and
       initiates all pins in array displayPins as outputs
104        pinMode(displayPins[i], OUTPUT);
105    }
106
107    for (int k = 0; k <= 3; k++) { //loops over and
       initiates all pins in array powerPins as outputs
108        pinMode(powerPins[k], OUTPUT);
109    }
110
111    Serial.println("program start");
112    attachInterrupt(interruptPin, trigger, RISING);
113

```

```

114 }
115
116 //program main-loop
117 void loop() {
118     //sub-loop for setting timer
119     digitalWrite(ledPin, 1);
120
121     while (true) {
122         currentTime = millis();    //update variable
123                                     //currentTime with new time readout in ms
124
125         switch(map(analogRead(potentiometerPin), 0, 1024, 0, 5))
126         {
127             case 0:
128                 timerLength = 0;
129                 tal[1][0] = 10;
130                 tal[1][1] = 10;
131                 tal[1][2] = 10;
132                 tal[1][3] = 0;
133                 break;
134
135             case 1:
136                 timerLength = 5000;
137                 tal[1][0] = 10;
138                 tal[1][1] = 10;
139                 tal[1][2] = 10;
140                 tal[1][3] = 5;
141                 break;
142
143             case 2:
144                 timerLength = 10000;
145                 tal[1][0] = 10;
146                 tal[1][1] = 10;
147                 tal[1][2] = 1;
148                 tal[1][3] = 0;
149                 break;
150
151             case 3:
152                 timerLength = 20000;
153                 tal[1][0] = 10;
154                 tal[1][1] = 10;
155                 tal[1][2] = 2;
156                 tal[1][3] = 0;
157                 break;
158
159             case 4:
160                 timerLength = 50000;
161                 tal[1][0] = 10;
162                 tal[1][1] = 10;
163                 tal[1][2] = 5;
164                 tal[1][3] = 0;
165                 break;
166
167             case 5:

```

```

166         timerLength = 90000;
167         tal[1][0] = 10;
168         tal[1][1] = 10;
169         tal[1][2] = 9;
170         tal[1][3] = 0;
171         break;
172     }
173
174     screenUpdate(1);
175
176     if (digitalRead(resetPin) == LOW && currentTime -
177         buttonPressed >= 250) { //moves forward in program if
178         buttonPressed = currentTime;
179         break;
180     }
181
182     digitalWrite(ledPin, 0);
183
184     timerAvilable = (timerLength != 0);
185
186     tal[0][0] = 10;
187     tal[0][1] = 10;
188     tal[0][2] = 10;
189     tal[0][3] = 0;
190
191     Serial.println("starting, timer set to:" + String(
192         timerLength));
193     counted = 0;
194     timerLength = currentTime + timerLength;
195
196     //sub-loop that counts hits
197     while (true) {
198         currentTime = millis(); //update variable
199         //currentTime with new time readout in ms
200
201         if (digitalRead(timeshowPin) == LOW) {
202             digitalWrite(ledPin, 1);
203             if (timerAvilable) {
204                 valueSplit(1, (((timerLength - currentTime) - ((
205                     timerLength - currentTime)%1000))/1000));
206             }
207             screenUpdate(1); //updates the next
208             //digit in the display if the previous one has been
209             //lit for 2 ms
210         }
211         else {
212             screenUpdate(0); //updates the next
213             //digit in the display if the previous one has been
214             //lit for 2 ms
215             digitalWrite(ledPin, 0);
216         }
217     }

```

```

212     if (timerAvilable == true && currentTime >= timerLength
213         || digitalRead(resetPin) == LOW && currentTime -
214             buttonPressed >= 250) {
215         buttonPressed = currentTime;
216         break;
217     }
218 }
219 Serial.println(" finished");
220 detachInterrupt(interruptPin);
221 digitalWrite(ledPin, 0);
222 //sub-loop that displays resulting counted hits
223 while (true) {
224     currentTime = millis(); //update variable
225                             //currentTime with new time readout in ms
226     if (currentTime%1000 <= 500) {
227         digitalWrite(powerPins[numberSSD - 1], 0); //
228             //updates the next digit in the display if the
229             //previous one has been lit for 2 ms
230     } else {
231         screenUpdate(0);
232     }
233     if (digitalRead(resetPin) == LOW && currentTime -
234         buttonPressed >= 250 ) {
235         buttonPressed = currentTime;
236         break;
237     }
238 }
239 attachInterrupt(interruptPin, trigger, RISING);

```