AARHUS UNIVERSITET

Aarhus University School of Engineering

BajerBuddy

4. Semesterprojekt

Gruppe 6

Vejleder: Søren H. Nielsen

Aleksander Wolter	Studienummer: 202005677	AU-id: au671571
Rodar Ahmd Rsol	Studienummer: 201911009	AU-id: au656587
Frederick Alexander Damgaard-Iversen Ulbæk	Studienummer: 202105438	AU-id: au699802
Maxim Kudelia Christiansen	Studienummer: 202107465	AU-id: au698924
Daniel Flemming Borch-Olsen	Studienummer: 201905293	AU-id: au632860
Justin Aidan Falk	Studienummer: 202106304	AU-id: au692557
Mads Evander Jensen	Studienummer: 202107547	AU-id: au705673

DATO

May 28, 2023

68.800 estimeret karakterer uden Resume/Abstract.
76.637 estimeret karakterer for hele dokumentet.

Nødstopsensor modultest

Modultesten for nødstopsensoren kan findes i bilag: [27].

13.3 Trådløs kommunikation

Radiofrekventkommunikation

Radiofrekvent kommunikation er en af formerne for trådløs kommunikation, som bruger radiobølger til at sende og modtage data. Der findes mange forskellige enheder som bruger radiokommunikation til kommunikation på de forskellige frekvenser. Radiobølger sendes via luft og kan bruges til at overfører forskellige typer data. Da radioantennen opsamler alle signaler på en gang, derfor er det vigtigt at den har et LC-kredsløb. [29]

ASK

Der findes mange forskellige metoder, som kan anvendes til trådløs kommunikation, grundet til at ask er blevet valgt, er denne metodes simpelhed. Denne modulationsmetode bruger forskellige amplituder af en bærebølge for at repræsentere digitale data. Det vil sige, at en høj amplitude svarer til et binær 1 og lav amplitude betyder et 0. Når der er tale om ASK er der generelt to typer synkron og asynkron. [30]

Synkron ASK funger ved tidsmæssigt synkroniseret dataoverførsel mellem modtager og sender. Dette betyder at dataene bliver sendt i faste tidsintervaller vha. den fælles tidsreference. Denne metode sikrer præcis koordinering, hvilket mindsker fejl i kommunikationen og påvirkning af støj. Asynkron ASK benytter sig derimod ikke af fælles tidsreference, hvilket betyder at hver bit bliver sendt uafhængigt. Denne metode er betydeligt nemmere at implementere, dog er den også mere tilbøjelig for fejl og påvirkning af støj.

FS1000A moduler

For at kunne transmittere og modtage data via radiofrekvent kommunikation, bruges FS1000A moduler. Disse moduler er designet til at kunne bruges på afstande op til 30-35 meter og er derfor godkendt ift. det ikke funktionelle must have krav nr. 11, som lyder: " Systemet skal kunne kommunikere med "targets" indenfor minimum 0-15 meters afstand". Senderen benytter sig af en SAW (Surface acoustic wave) resonator på 433 MHz til at transmittere radiobølger [5]. Modtager modulet er lidt mere avanceret da det gør brug af en operationsforstærker til at forstærke den modtagende radiobølge, hvor signalet herefter sendes til en "phase lock loop" (PLL). PPL-delen synkroniserer outputfrekvensen og fasen med indgangssignalet. Via en feedbackmekanisme og en spændingsstyret oscillator kan en sekvens af digitale bits afkodes og sendes ud via data pinen [31].

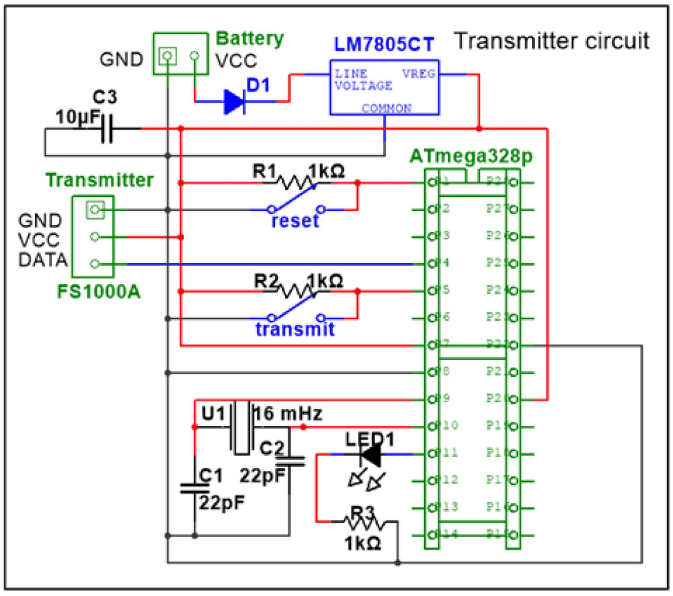
Sender modul:
Pin 1: Data
Pin 2: 5V VCC
Pin 3: GND

Modtager modul:
Pin 1: 5V VCC
Pin 2 & 3: Data
Pin 4: GND

Figur 62: FS1000A moduler([5])

HW - Sender

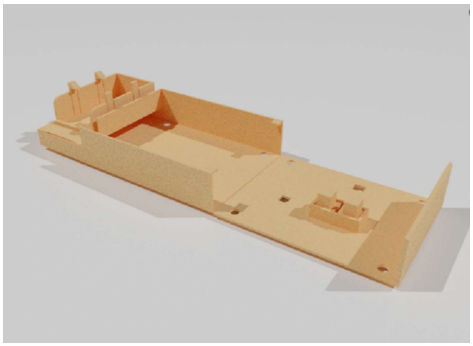
Da vores system bajerbuddy skal kunne blive tilkaldt af kunder der sidder ved forskellige borde, skal vi have nogle knapper på bordene der sender et unikt signal til bajerbuddy. Her bruger vi den trådløse kommunikationsmetode ASK over radiobølger. Da vi allerede har en sender og modtagere på lager bruger vi dem. Det er type FS1000A sender og modtager, der er lav budget RF sender – modtager moduler. De er simple moduler der sender ved en meget høj frekvens, i vores tilfælde 433 MHz. Det eneste vi skal gøre, er at designe et kredsløb der kan sende ASK koden på FS1000A modulets data pin. For at sende koden bruger vi et ASK bibliotek der konverterer en streng til en bit-sekvens vores FS1000A moduler kan sende og modtage. Til dette skal vi bruge en mikrokontroller, og da vi har meget erfaring med Arduino Uno fra studiet bruger vi ATmega328p chippen. Herunder ses kredsløbet vi er kommet op med.



Figur 63: kredsløbstegning

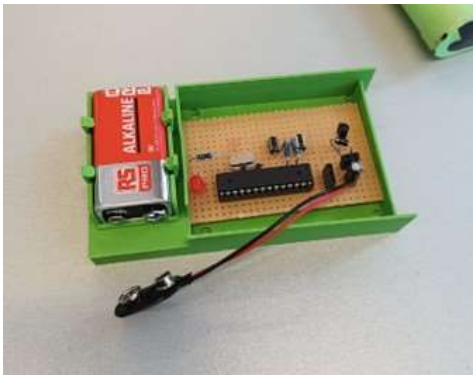
Fra toppen er vores indgang fra et batteri. Vi har på dette tidspunkt ikke fundet en løsning til batteriet så vi designede kredsløbet med en spændings regulator der altid sænker spændingen til 5 V. Vi bruger også en diode da vi helt vil undgå at kortslutte mikroprocessoren. Vi har en 10µF kondensator for at sørge for at mikroprocessoren altid har nok strøm. Derunder har vi to knapper, den ene til at genstarte processoren og den anden til at sende vores ASK kode. Vores AF1000A sider på boardet og har kun 3 indgange, stel, VCC og en data pin. Data pin'en er forbundet direkte til en digital pin på mikroprocessoren. Sidst har vi en krystal der kører processoren og to nødvendige kondensatorer fra databladet [32] og ved siden af en enkelt LED der både fungere som feedback når man trykker på knappen og også har fungeret som en debug LED. Alt dette er hvad transmitteren til bajerbuddy skal bruge. Den unikke ASK kode er hardcoded ind i mikroprocessoren.

Vi har valgt at designet kabinettet til at kunne holde et 9V batteri. Designet og den 3d printede version ses herunder. Designet er lavet i blender.



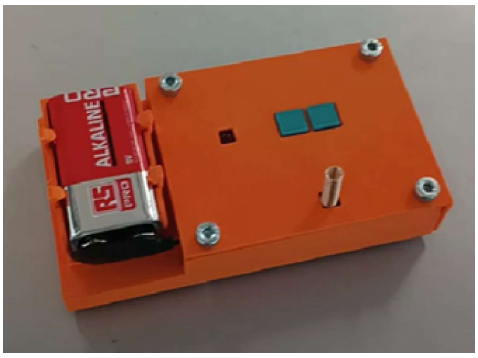
Figur 64: Transmitter kabinet design

Første prototype.



Figur 65: Transmitter 3d print

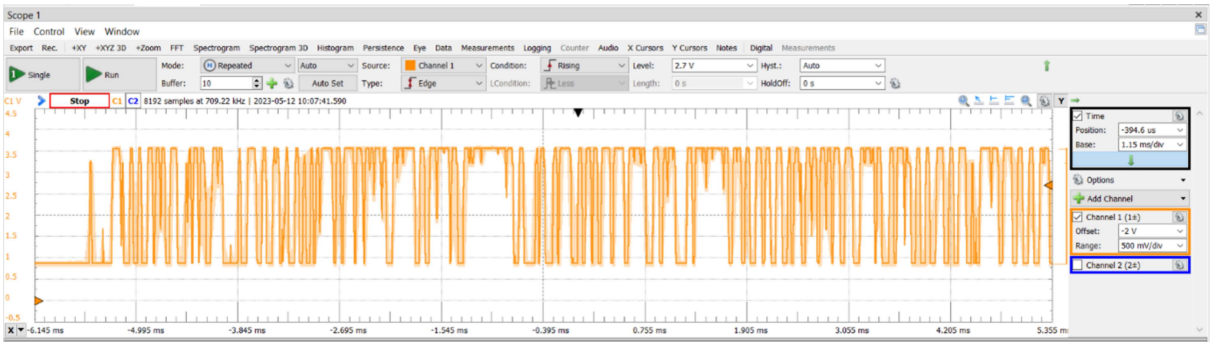
Anden prototype, med knapper og låg.



Figur 66: Transmitter prototype

HW - Modtager

Modtager modulet som er del af Bajerbuddy's hovedsystem er designet til at modtage information fra de sender moduler som, befinder sig på bordene. Modtagerens formål er at oversætte data fra et sendermodul til det bord som Bajerbuddy skal navigere over til når den er klar. Dette gøres ved at bruge et FS1000A 433MHz RF modtager modul. Modtager modulet består af et FS1000A receiver modul, MCP601 operationsforstærker og et lavpas filter. Lavpas filteret har været nødvendigt at montere da der var store mængder støj på FS1000A modulets data pin (pin 2), som ses på følgende oscilloskop billede:



Figur 67: Støj på data pin (pin2), når der ikke sendes data

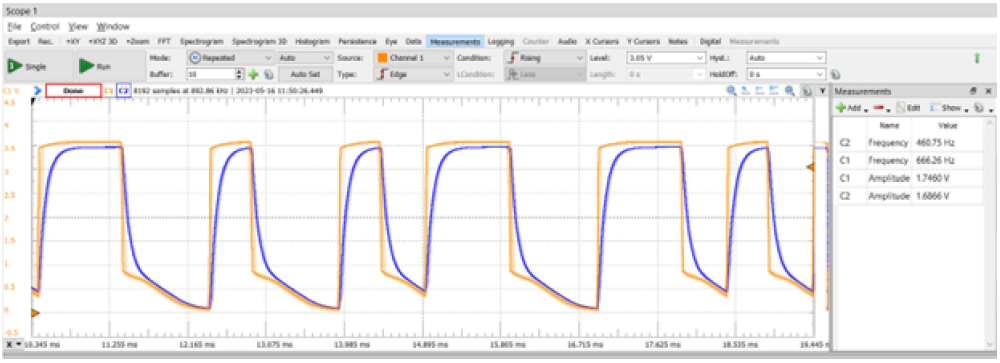
Det ses på grafen at der store mængder støj, hvilket kan komme fra mange forskellige kilder. Dette kan skyldes at 433 MHz ikke er en af de beskyttede radiofrekvenser og derfor er det nødvendigt at filtrere outputtet [33]. Der dimensioneres et lavpas filter med en knækfrekvens på 600 Hz, da det vides at når FS1000A modtager noget data er output signalet på omkring 500 Hz.
Dimensioneringen af lavpas filtret ses her:

$$\frac{1}{2 \cdot \pi \cdot fc \cdot C} = R$$

$$\frac{1}{2 \cdot \pi \cdot 600 \cdot 47 \cdot 10^{-9}} = 5.644 \cdot 10^{3}$$

Figur 68: Dimensionering af RC-led

Det vides nu at kondensatoren er 47 nano farad og en modstand på ca. 5,7 kiloohm og ledet kan derfor implementeres. Output signalet undersøges nu via oscilloskop (gul er inden filter og blå er efter), hvilket ses på nedenstående billede:



Figur 69: Output data før og efter lavpass filter

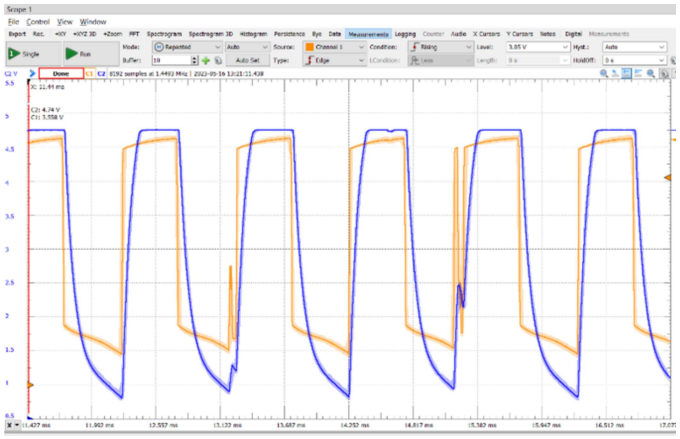
Da signalets støj nu er dæmpet, skal det forstrækkes for at en RPI kan aflæse signalet. En RPI kan først læste et logisk højt signal når det er mellem 1.8 V-3.3V og derfor tilføjes en operationsforstærker [34]. Det vides at signalet inden operationsforstærkeren er 1.66V, hvilket ikke er nok. Signalet forstærkes kun 1.5 gang da signalet heller 3.3 V.

$$Rf := 500\ \Omega \qquad Rin := 1000\ \Omega \qquad V\_in := 1.66\ V$$

$$V\_out := \left(1 + \frac{Rf}{Rin}\right) \cdot V\_in = 2.49\ V$$

Figur 70: Dimensionering af forstærker

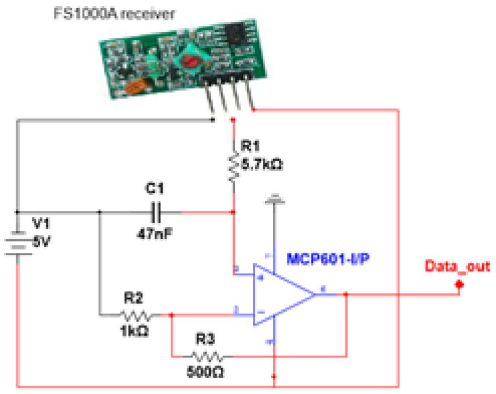
Efter at operationsforstærkeren er tilføjet undersøges output signalet igen via oscilloskop:



Figur 71: Output data før lavpass filter & forstærkning (gul) og efter (blå)

Det ses nu at signalet er over 1.8 V og det kan dermed aflæses af RPI. Et diagram for det

færdigdesignede kredsløbet ses nedenfor:



Figur 72: Kredsløbsdiagram for modtager modul

SW - Sender

Herunder ses koden til transmitteren. Vi bruger ASK biblioteket 'RadioHead', her kaldet 'RH_ASK.h' . Driveren bliver initialiseret med RH-_ASK klassen. I vores tilfælde sender vi med 2000 bit/sek på pin 13 på mikrocontrolleren. I bunden af koden ses hvordan vi tjekker om knappen bliver trykket ned, og hvis den gør, bruger vi ASK driveren til at sende beskeden. Beskeden gemt i variablen 'msg' kan vi ændre på hver mikrochip så den sender en unik besked. Koden til senderen findes i billag [35].

```
#include <RH_ASK.h>
#ifdef RH_HAVE_HARDWARE_SPI
#include <SPI.h>
#endif

RH_ASK driver(2000, 13, 13, 0);

const int buttonPin = 4;
int buttonState = 0;

void setup()
{
  pinMode(buttonPin, INPUT);

  #ifdef RH_HAVE_SERIAL
      Serial.begin(9600);
  #endif
      if (!driver.init())
  #ifdef RH_HAVE_SERIAL
          Serial.println("init failed");
  #else
  ;
#endif
}

const char msg = "432";

void loop()
{
  buttonState = digitalRead(buttonPin);

  if (buttonState == HIGH) {
    Serial.println("button pressed");
    driver.send((uint8_t)msg, strlen(msg));
    driver.waitPacketSent();
    delay(100);
  }
}
```

Figur 73: kredsløbstegning

SW - Modtager

Da sender modulen benytter sig af RadioHead bibliotek til at sende data vha. ASK metoden, og RadioHead biblioteket er skrevet specifikt til Arduino, var det ikke muligt at koble 433MHz modtager modulet direkte til RPI pga. den korte tidshorisont og andre prioriteringer. Derfor blev der også brugt en Arduino på modtager modul som mellemled mellem modtageren og RPI. Før denne løsning blev implementeret, blev der set på alternative løsninger som kunne undvære den ekstra Arduino. Den første mulighed kunne have været at anvende den eksisterende klasse i RadioHead bibliotek til RPI, ved navnet "RH_RF24" [36]. Dog kommuniker denne klasse med modtager modul via SPI, og da vores valgte modtager kun kommuniker via TTL, kan denne løsning ikke implementeres medmindre hardwaremodulet udskiftes med en SPI kompatibel modul, som fx Sparkfun WRL-00691 [37]. Dog pga. manglende tilgængelighed og prisen blev dette modul ikke implementeret. Et alternativ som blev forsøgt, er at omskrive RadioHead bibliotek. Da modtager modulet sender et TTL-signal på dens data pin, blev der forsøgt at implementerer nogen simple funktioner, der kunne via " digitalRead()" funktionen fra WiringPi-biblioteket afkode den modtaget data.

```
51  // Funktion til at skubbe et nyt bit ind i bufferet
52  void shiftBuffer(int buffer[], int bit) {
53      for (int i = BUFFER_SIZE - 1; i > 0; i--) {
54          buffer[i] = buffer[i - 1];
55      }
56      buffer[0] = bit;
57  }
58
59  // Funktion til at kontrollere om bufferen er fyldt med BITSIZE bits
60  int isBufferFull(int buffer[]) {
61      for (int i = 0; i < BUFFER_SIZE; i++) {
62          if (buffer[i] == -1) {
63              return 0;
64          }
65      }
66      return 1;
67  }
68
69  // Funktion til at konvertere bufferen til en byte
70  int convertBufferToByte(int buffer[]) {
71      int byte = 0;
72      for (int i = 0; i < BUFFER_SIZE; i++) {
73          byte = (byte << 1) | buffer[i];
74      }
75      return byte;
76  }
77
78  // Funktion til at nulstille bufferen
79  void resetBuffer(int buffer[]) {
80      for (int i = 0; i < BUFFER_SIZE; i++) {
81          buffer[i] = -1;
82      }
83  }
```

Figur 74: Forsøg på omskrivning af RadioHead

Det lykkedes ikke at implementere disse funktioner af uvisse fejl. Et bud på hvordan det kunne

være, er der mangler fejlhåndtering i koden og da der bruges den asynkrone metode som er upålidelig og med stor sandsynlighed førte til tab af data. Da disse to løsninger ikke var mulige at implementere, blev der lavet en alternativ løsning med en Arduino som mellemled. Denne løsninger bruger RadioHead klassen RH_ASK. Der bliver advent følgende kode, som konfigurerer RF-moduler og opretter en seriel kommunikationsforbindelse til en Raspberry Pi ved hjælp af bibliotekerne RH_ASK og SoftwareSerial. Den fortsætter med at modtage data kontinuerligt via RF-modulerne og sender dem derefter videre til Raspberry Pi gennem den serielle forbindelse udklip af koden kan ses på nedrestående figur. Hele koden findes i bilag [38]

```
22  void loop() {
23      uint8_t buf[RH_ASK_MAX_MESSAGE_LEN];
24      uint8_t buflen = sizeof(buf);
25
26      if (driver.recv(buf, &buflen)) // Non-blocking
27      {
28          // Besked modtaget, send den via UART til Raspberry Pi
29          for (int i = 0; i < buflen; i++) {
30              serialPi.write(buf[i]);
31  #ifdef RH_HAVE_SERIAL
32              Serial.print((char)buf[i]);
33  #endif
34          }
35  #ifdef RH_HAVE_SERIAL
36          Serial.println();
37  #endif
38      }
39  }
```

Figur 75: Udklip af Arduino modtager koden

Sidste led i modtager modulet er rf-mod klassen, som er blevet skrevet til RPI, for at den kan modtage UART kommunikation fra Arduinoen. Da vores mikrocontroller kører på et image fra ISU/ HAL undervisning kræver det en lille ændring i cmdline.txt filen før UART kan læses på "ttyS0". Følgende linje skulle skiftes i den nævnte fil.

```
dwc_otg.lpm_enable=0 root=/dev/mmcblk0p2 rootfstype=ext4 rootwait modules-load=dwc2,g_ether
```

Den fulde vejledning kunne findes på Raspberry Zero Wiki fra HAL undervisning. [39]

```
public:
    SerialReceiver() {
        if (wiringPiSetup() == -1) {
            std::cout << "Kunne ikke initialisere WiringPi-biblioteket. Afslutter programmet." << std::endl;
            return;
        }

        if ((serialPort = serialOpen("/dev/ttyS0", 9600)) < 0) {
            std::cout << "Kunne ikke åbne serielporten. Afslutter programmet." << std::endl;
            return;
        }
    }

    ~SerialReceiver() {
        serialClose(serialPort);
    }

    void receiveData() {
        std::string receivedData;

        while (true) {
            if (serialDataAvail(serialPort)) {
                char c = serialGetchar(serialPort);

                if (c != '\n' && c != '\r') {
                    receivedData += c;
                } else if (!receivedData.empty()) {
                    processData(receivedData);
                    receivedData.clear();
                }
            }
        }
    }

private:
    int serialPort;

    void processData(const std::string& data) {
        std::cout << "Modtaget data: " << data << std::endl;
    }
};
```

Figur 76: rf-mod klassen

På overstående figur 76 ses klassen til modtagen af UART. Klassen bruger WiringPi-biblioteket til at modtage data fra en seriel port. Den kontinuerligt modtager data og gemmer dem i en streng. Når en fuld besked er modtaget, bliver den behandlet ved at udskrive den modtagne data. Koden til modtager findes i billag [40].

13.4 Modul-test - Sender og modtagert

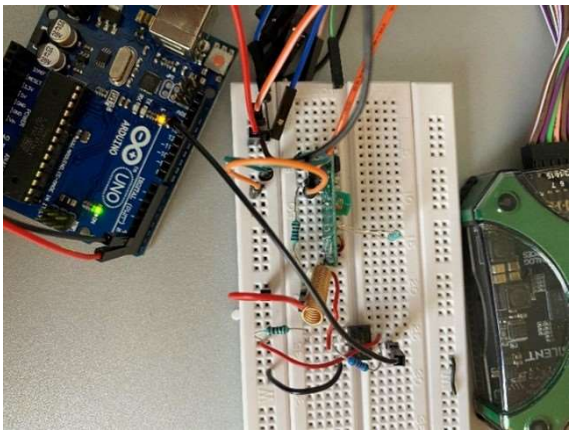
Under modul-test blev der først set på senderen. Sendermodulet sender via 350 MHz, hvilket gør det ikke muligt at analyser det sendte signal med labs oscilloskoper, da de kun kan måle op til 70MHz. Universitetet har også højfrekvente oscilloskoper, som kan måle op til flere gigahertz, dog blev det ikke muligt at teste sender modulet på et højfrekvente oscilloskop, da testfasen startede sent og det var ikke muligt at få adgang til et højfrekvente oscilloskop i det korte tidsrum. Vi starter med at kigge på senderen. Det ses herunder at når der bliver trykket på sende modulets knap, at der bliver sendt data.

Figur 77: Transmitter data

Vi tester modtageren på samme vis, vi måler på data pin'en og kan se at signalet er kommet i gennem og altså er blevet sendt mellem RF modulerne.

Figur 78: Modtager læser 1 og 0

På følgende figur kan der ses en af test opstillinger, hvor sender modulet samt lavpasfilteret med forstærkeren blev testet, før modulet blev loddet på veroboard.



Figur 79: Testopstilling af modtagermodul

Når vi afkoder signalet med ASK biblioteket kan vi se at den rigtige unikke kode bliver sendt ud.

```
root@raspberrypi0-wifi:~# ./rf-mod
Modtaget data: bordXYZ
Modtaget data: bordXYZ
Modtaget data: bordXYZ
Modtaget data: bordXYZ
Modtaget data: bordXYZ
Modtaget data: bordXYZ
```

Figur 80: Terminal udskrift fra RPI

Vi har dog lagt mærke til at RF-modulerne ikke kan sende særligt langt. Vi kan dog justere på modtagerens frekvens ved at skrue på en variabel induktor. Når vi justerer den, er det længste vi kan sende omkring 1 meter. Dette er meget ærgerligt da vi håbede på at kunne sende i hvert fald 10 meter, men det er bare en begrænsning af de små RF-moduler vi har til rådighed og der vurderes at det er tilfredsstillende til vores prototype.

Batteritid

Vi har målt senderen til at bruge 7.7 mA i idle og 11.5 mA når der bliver sendt data. Et normalt 9V batteri har 500 mAh. Hvilket betyder systemet kan køre uafbrudt i 65 timer i idle, det er alt for dårligt. Vi kunne lave nogle forbedringer for at sænke strømforbruget. F.eks, er det meget simpelt at bruge en krystal med lavere frekvens. Når man sænker clock frekvensen af chippen kan man også sænke den spænding der skal til for at køre den. Derudover kunne vi også optimere koden, så chippen går i 'power-down mode' indtil knappen bliver trykket.