Projekt Larm

Innehållsförteckning

[Innehållsförteckning 1](#_Toc480470464)

[Inledning 2](#_Toc480470465)

[1.1 Bakgrund 2](#_Toc480470466)

[1.2 syfte 2](#_Toc480470467)

[1.2 Källor 2](#_Toc480470468)

[Larmfunktionalitet 3](#_Toc480470469)

[Första funktionalitetstanken 3](#_Toc480470470)

[Steg 1: 3](#_Toc480470471)

[Steg 2: 3](#_Toc480470472)

[Steg 3: 3](#_Toc480470473)

[Steg 4: 3](#_Toc480470474)

[Steg 5: 3](#_Toc480470475)

[Första tänkta flödesschemat 4](#_Toc480470476)

[Flödesschema Hiss Stefan Ekström och Marcus Arvidsson 4](#_Toc480470477)

[Nuvarande funktionalitet 5](#_Toc480470478)

[Snabb funktionalitet sammanfattning 5](#_Toc480470479)

[Teori 6](#_Toc480470480)

[Moduler, hårdvara och produkten 7](#_Toc480470481)

[Kod 10](#_Toc480470482)

[Slutsatser 16](#_Toc480470483)

[Har gått bra 16](#_Toc480470484)

[Har gått dåligt 16](#_Toc480470485)

Inledning

### Bakgrund

Peter Björkholm på BFC har beställt ett rörelselarm ifrån oss . Ett rörelselarm med specifika funktioner som vi måste uppfylla för att kunden ska bli nöjd. Har fått en kravspecifikation som ska uppfyllas innan den blir godkänt för utskick till kund.

### 1.2 syfte

Rörelselarmets huvuduppgifter är att larma. Om någon eller något rör sig framför en sensor så kommer ett larm att tjuta, lampa lysa och larmet kommer även upplysa larmägaren via SMS att ägarens larm har aktiverats eller blivit utlöst. Ett projekt med komponentlista, schema och flödesschema. Men följa kravspecifikationen så gott det går och allt behöver inte vara uppfyllt för den ska bli godkänd.

### Källor

Under projektets gång har handledare funnits till hands för rådgivning. Även har internet varit till största hjälpen inför detta projekt. Manualer, faktablad och olika andra projekt från människor med öppen kod för allmänheten att använda sig utav.

## Larmfunktionalitet

### Första funktionalitetstanken

Tanken med detta projekt är att vi vill skapa en produkt som är portabel, Ett larm med siren och rörelse sensor som går på batterier. Vi börjar med att enkelt få en lampa att lysa. Det vi vill med projektet är att kan använda oss även av kamera/ta bild/filma och skicka till SD kort och hemsida. Vi vill även kunna stänga av larmet fysikt genom att skicka ett sms till modulen.

Vi har kommit överens om att vår nivå ska ligga på 3-4 och om vi har möjlighet så kommer vi jobba oss uppåt för att nå en högre ambitionsnivå. Kanske även en kommersiell produkt. De viktigaste är för oss att inte ta vatten över huvudet, bättre att vi börjar lägre och få en produkt som fungerar än att vi står där med en icke fungerande produkt.

### Steg 1:

* Fungerar rörelse sensorn? Känner den av något alls? Lampa tänds för att kunna kontrollera att de fungerar röd/grön lampa.

### Steg 2:

* Efter igenkäninng börjar vi skriva beskriva vad sensorn ska göra (programkod)

### Steg 3:

* Lägga till ljud och ljussignal

### Steg 4:

* Göra så att den loggar till SD kort/html fil

### Steg 5:

* Tillämpa sms funktion så att den skickar sms när någon är i huset/lägenheten.

Som ni kommer att se i nuvarande funktionalitet nedan så har vi inte någon kamera, HTML sida, SD kort eller GSM. Man kan inte stänga av den via mobil utan bara få avisering till mottagaren.

### Första tänkta flödesschemat

# Flödesschema Hiss Stefan Ekström och Marcus Arvidsson

Stanna

Våning 2

Våning 1

Röd

Alarm

Gul 2

Gul1

Grön2

Grön 1

Nödstppp

V2inne

V1inne

V2ute

V1ute

### Nuvarande funktionalitet

Första stadiet när man kopplat i larmet med kod är att det är inte beroende av stadiet som larmet var på senast den användes, aktiverad eller avaktiverad utan snarare den är alltid avaktiverad vid start, när spänning sätts på. Om rörelsesensorn är aktiverad så kommer den aktivera rörelsesensorn som är kopplad till en röd lampa och larmsiren. Går någon eller något förbi vår rörelsesensor så aktiveras ljudet och kommer tjuta tills larmet stängs av. Sedan så stänger man av larmet med vår pinkod och då inaktiveras det och den röda lampan slutar att lysa och sedan börjar en grön lampa lysa som säger att larmet är avaktiverat. Vill man sedan aktivera larmet igen så skriver man samma pinkod 1,2,3,4 plus A (aktivering) och om man vill inaktivera larmet skriver man 1,2,3,4 plus D (Deaktivering).

Så här gör vi: i loopen detekterar ni knapptryckningen med

char customKey = customKeypad.getKey();

Som vi gör nu. Vi måste också ha en array som sparar föregående knapptryckningar, och vi måste se till att lagra knapptryckningarna i ordning så att första knapptryckningen som vi har sparat är '1', andra är '2' osv.. När vi kommer till femte knapptryckningen och den är 'A' och arrayen innehåller '1', '2', '3' och '4', så slår vi på sensorn för aktivering, om den är 'D' och arrayen innehåller '1', '2', '3' och '4' så slår vi av sensorn, om vi får något annat, så nollställer vi arrayen och det kommer att stå ”Fel kod” med blinkande röd lampa och man får börja om med pinkoden. Vi måste även ha en variabel som känner av ifall den i sitt senaste stadie antingen var aktiverad eller avaktiverad så man snabbt kan testa larmet men har uteslutit denna funktion helt.

### Snabb funktionalitet sammanfattning

Vårat larm är alltid avstängt vid start. Man måste aktivera det med våran pinkod 1, 2, 3, 4, och ’A’ och då startar en rörelsesensor som antingen känner av någonting eller inte. Känner den inte av någonting så kommer den inte göra någonting men om den gör det så kommer den gå in i ett annat villkor som betyder att om den känner av något så ska jag starta en röd lampa först och sedan några sekunder senare börja tjuta från en siren. Då kommer den ligga där med en fördröjning och tjuta tills den fördröjningen är slut så länge den inte känner att någon är i rummet. För att stänga av larmet måste vi använda oss av samma pinkod fast så lägger vi till ’D’ för deaktivering istället för ’A’ för aktivering. Då kommer larmet sluta tjuta, röd lampa släckas och sedan även kommer en grön lampa börja lysa som säger att larmet är avstängt och väntar på aktivering. Skriver man in fel kod för deaktivering eller aktivering så kommer röda lampan blinka.

## Teori

Som det funkar i teorin funkar nästan aldrig i praktiken, men det är så nära man kan komma ändå. Vi använder oss utav olika komponenter som talar med en MCU. Men även i inbyggda system är det otroligt viktigt att att man använder rätt spänningar med och veta hur länge som battrierna kommer att hålla realtid i sitt inbyggda system innan man behöver deaktivara sitt kodlarm och byta ut dem hos antingen kunden eller att de klarar de själva.

I vårat inbyggda system som är ett såkallat rörelselarm behövs inte mer än 5V egentligen men vi har tweakat det med lite högre spänning på en volt mer bara. Så vårat system går på 6V och därmed så är det exakt fyra stycken 1.5V AA batterier som driver hela systemet och produkten. Även om sirenen klarar 12V så kommer den låta högt.

Systemet driver inte allt samtidigt för då skulle troligen systemet krascha pga alldeles för lite matning. I vårat system har vi en klart en MCU(ATMEGA arduino 2560) heter den och är en större version utav Arduino Uno som är hjärnan som kopplar ihop alla vår komponenter och kod så de talar med varandra på rätt sätt som vi vill att den ska tala. Men troligtvis teoretiskt sätt skulle kanske denna produkt som den ser ut idag kunna köras bara på en arduino UNO.

Kopplat till MCU:n har vi två stycken resistorer på 330 ohm, två lysdioder(grön och röd), pinkodsläsare, rörelsesensor och en tjutande siren(larm sirenen). Närmare på vad som exakt används kommer på nästa sida **moduler, hårdvara och produkten.**

## Moduler, hårdvara och produkten

Som i alla inbyggda system finns det kod och så finns det hårdvara, så kallade elektriska komponenter med olika funktioner, metaller som ledare, transistorer som leder strömmen rätt, isolatorer och isolatorer som reglerar strömmstyrkorna, I/O:n som vi satt komponenterna i för att tala med kortet och bussar som som man överför kod med o.s.v. Olika protokoll du nämner det.

I vårat system har vi som sagt olika komponenter med. Och här visas vilka komponenter vi har använt oss utav.

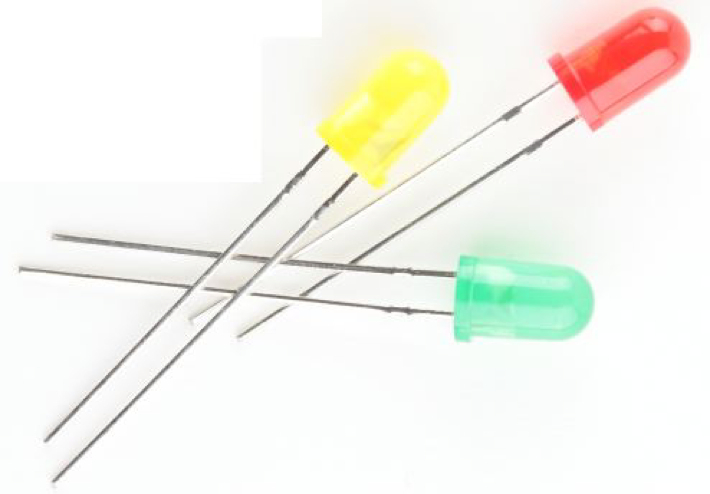
**MCU:** ATMEGA 2560 arduino. Hjärnan som kopplar kod och komponenter så de kan tala med varandra. Mikroprocessor enhet som är hjärnan i produkten



**Keypad matrix:** En pinkodsläsare som initieras med ett bibliotek i arduino miljön där man kan definera dess knappar med hjälp av två dimensionella fält och bestämma exakt vad man vill ska hända när man trycker på dess knappar.



**Lysdioder:** Grön och röd med två egna digitala utgångar som man kan reglera själv i koden man gjord för hela produkten. Vanliga enkla lysdioder för enkel igenkänning vid olika funktioner man enkelt kan se med ögonen att något händer när man trycker på pinläsaren.



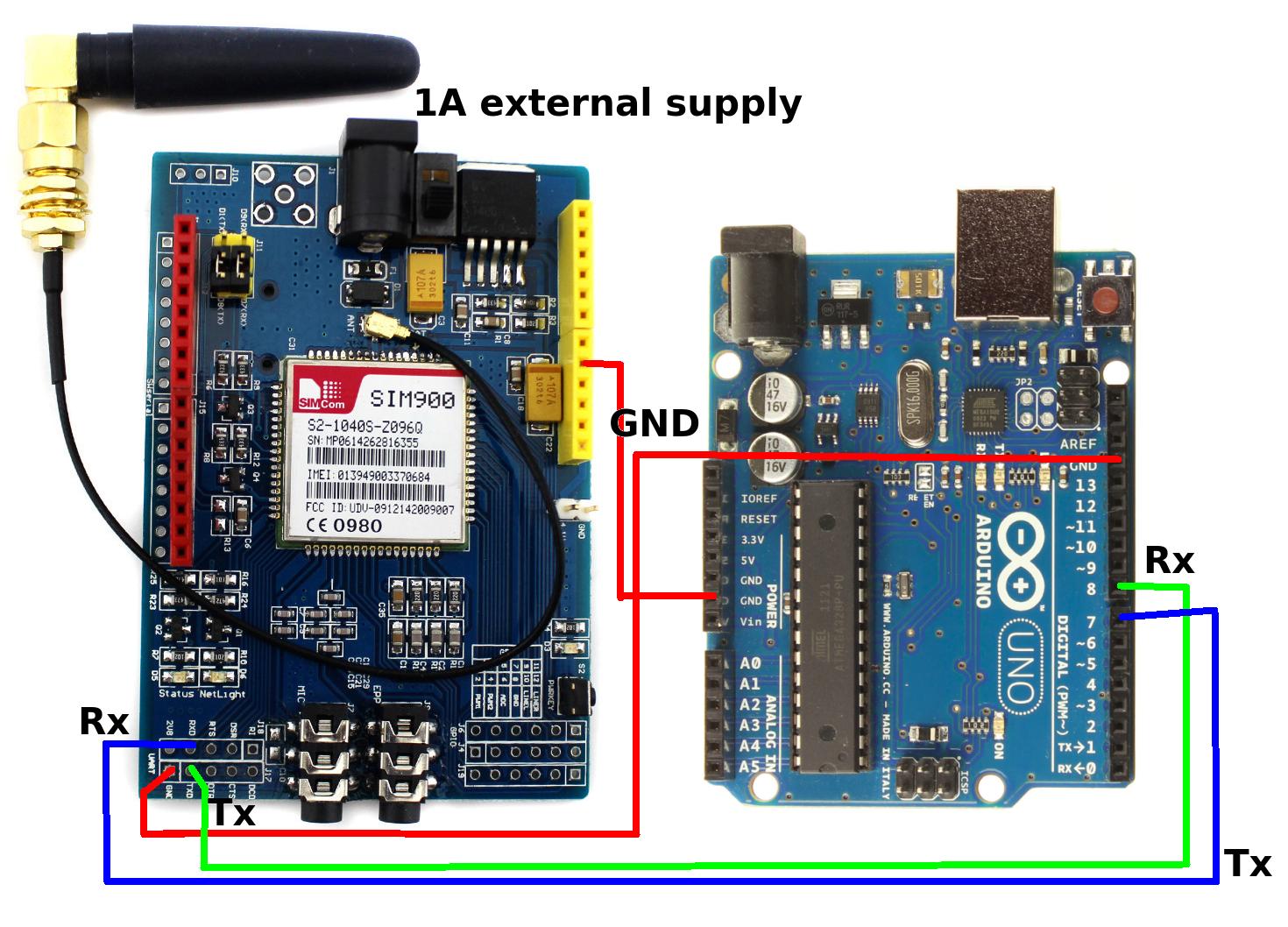
**Summer 75dB siren:** En siren som tjuter högt när någon rör sig omkring våran rörelse sensor.



**PIR rörelsesensor:** Våran rörelsesensor som känner av ifall någon är i dess närhet och om nu då vårat larm är aktiverat så kommer den att reagera och utlösa vårat larm.



**Simkortsmodul SIM 800L:** Denna modul fanns med i våran projekt men inte i kravspecifikationen. Dess jobb var att skicka ett SMS till larmets ägares mobiltelefon, men detta han man inte med innan det skulle vara klart. I **kod** sidan längre ner kommer en exempel kod på hur det kunnat fungera. Modulen till vänster om arduino UNO:n.



# Kod

#include <Keypad.h>

#include <SoftwareSerial.h>

int ledgreen=13; // Grön lampa

int msensor1=12; // Rörelsesensor

int ledred=11; // Röd lampa

int sound = 10; // Siren

char password\_on[] = {'1','2','3','4','A'}; //Koden för att aktivera larmet

char password\_off[] = {'1','2','3','4','D'}; //Koden för att inaktivera larmet

char keypressed[5];

bool kod\_on\_ok = true;

//bool status\_on = false;

int kodsiffra = 0; //Talar om vilken plats i password-arrayerna vi befinner oss på

/\* --------------- KOD FÖR ATT INITIERA KEYPAD ----------------------\*/

const byte ROWS = 4; //fyra rader

const byte COLS = 4; //fyra kolumner

//Definerar keypads värden på knapparna i en två dimensionellt fält.

char hexaKeys[ROWS][COLS] = {

{'1','2','3','A'},

{'4','5','6','B'},

{'7','8','9','C'},

{'\*','0','#','D'}

};

byte rowPins[ROWS] = {9, 8, 7, 6}; // ORANGE, BLÅ, BLÅ ,BLÅ //färger på kablarna

byte colPins[COLS] = {5, 4, 3, 2}; // GRÖN, GRÖN, GRÖN, GUL // färger på kablarna

Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

/\* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*/

/\* Vår sensor är aktiv \*/

void alarm\_on()

{

Serial.println("ALARM PÅ");

if(digitalRead(msensor1)==HIGH){

// msensor1 = HIGH;

digitalWrite(ledgreen, LOW);

delay(1000);

digitalWrite(ledred, HIGH);

delay(2000); // Senare ska detta vara 20 sekunder och GSM före.

digitalWrite(sound, HIGH);

}

else if(digitalRead(msensor1)==LOW){

// msensor1 = LOW;

digitalWrite(ledred, LOW);

delay(2000);

digitalWrite(sound, LOW);

}

}

/\* Vår sensor är inaktiv \*/

void alarm\_off()

{

msensor1 = LOW;

Serial.println("ALARM AV");

delay(1000);

digitalWrite(ledgreen, HIGH);

delay(1000);

digitalWrite(ledred, LOW);

delay(1000);

digitalWrite(sound, LOW);

}

void setup() {

Serial.begin(9600);

pinMode(ledgreen,OUTPUT);

pinMode(ledred,OUTPUT);

pinMode(msensor1,INPUT);

pinMode(sound,OUTPUT);

}

void loop() {

char customKey = customKeypad.getKey();

if (customKey) //Tittar och ser vad du trycker på pin knapparna.

{

keypressed[kodsiffra] = customKey;//Med villkor att kodsiffran ökar så länge som du inte trycker över 5 knappar.

kodsiffra++;

if (kodsiffra >= 5)//Här bestämmer den max antal knappar du kan trycka.

{

bool kod\_on\_ok = true;

//Kollar siffra för siffra och ser om den överensstämmer med vår kod för aktivering.

for (int i = 0; i <5; i++)

{

if (keypressed[i] != password\_on[i])

kod\_on\_ok = false;

}

/\*Om on - koden är sann så kallar han på alarm\_on funktionen.\*/

if (kod\_on\_ok == true){

alarm\_on();//Kallar på alarm\_on funktion.

}

bool kod\_off\_ok = true;

//Kollar siffra för siffra och ser om den överensstämmer med vår kod för deaktivering

for (int i = 0; i <5; i++)

{

if (keypressed[i] != password\_off[i])

kod\_off\_ok = false;

}

/\*Om deaktiveringskoden är sann\*/

if (kod\_off\_ok == true)

{

alarm\_off();//Kallar på alarm\_off funktionen.

}

/\*Kod för hantering utav fel inmatade pins. Om båda är falska så

så kommer den att säga det är fel och även blinka rött att du tryckt in fel kod

för att enkelt se det.\*/

kodsiffra = 0;

if ((kod\_on\_ok == false) && (kod\_off\_ok == false))

{

Serial.println("FEL KOD!");

digitalWrite(ledred , HIGH);

delay(1000);

digitalWrite(ledred , LOW);

delay(1000);

digitalWrite(ledred , HIGH);

delay(1000);

digitalWrite(ledred , LOW);

delay(1000);

digitalWrite(ledred , HIGH);

delay(1000);

digitalWrite(ledred , LOW);

}

}

}

}

## Slutsatser

Ett roligt projekt vårat juniorteam fått i uppgift och lärorikt till att säga det minsta. Saker som har gått bra och dåligt tar vi upp. Projektet har haft sina upp- och nedgångar men framförallt har det åtminstone gått frammåt. Det är svårt att jobba med billig teknik också då det haverarar enkelt, sitter dåligt fast, ologiska anslutningar o.s.v. Kan ej förvänta sig allt för mycket.

### Har gått bra

När vi väl fick heldagar att sitta med projektet så gick det inget annat än frammåt. Önskade bara vi kunde implementera allt vi ville innan det var dags för dess deadline. Bra hjälp från lärare, datablad och internet.

### Har gått dåligt

För de första så hade vi inte grejjorna från start utan fick beställa in komponenterna själva och de förlorar vi tid på men sånt ska man alltid räkna in i tiden. Sjukdomar och föräldraledigt har även drabbat projektgruppen och blir även du oflyt på projektet. Sådant man även där får räkna in men annars har vi bara oss själva att skylla. Men allt som allt ett roligt projekt.