Mazen Allaou, Dennis Bunne och Marcus Arvidsson

Epic larmsystem

Projektspecifikation

Ett larm som går att aktivera manuellt men även automatiskt beroende på vilka tider som larmet är inställt på att aktiveras/avaktiveras. Larm som jobbar inom olika tidsintervall med en realtidsklocka som är ihopkopplad med larmet.

Deltagare

- Dennis Bunne
- Mazen Allaou
- Marcus Arvidsson

Rollfördelning

I projektet har rollerna inte helt bestämts, vi alla har jobbat med samma sak oftast förutom i slutet då vi har delat upp arbetet för att göra klart projektet i tid.

I systemutvecklingen har det enbart varit ren parprogrammering där vi satt runt en dator och kodade tillsammans, för att enklare kunna lösa olika problem men även för att implementera nya idéer som skulle kunna göra larmet smartare och stabilare. Gruppsammankallningen har ingen stått för utom att var och en sköter sig själv. Däremot hade vi samma tider att passa varje gång vi skulle arbeta.

Användning

Vårt larmsystem är ett realtids larm. Det är ett larm som är inom drift dygnet runt och följer realtids klockans tidsintervaller. Larmet kommer att jobba inom olika intervaller för vardagar och helger. Detta sker automatiskt inom olika tidsramar. Helgerna är larmet på hela dygnet fram till Måndag morgonen klockan 08.00 och sedan tar ett annat vardags intervall på realtidsklockan över och håller larmet avstängt mellan 08.00 och 17.00 för att sedan sättas på igen efter 17.00 och 08.00 stängas av igen.

Larmet går även att aktivera och avaktivera manuellt med hjälp av pin kod på en keypad. Dess prioritet är högre än tidsintervallen men tas över sedan utav realtidsklockan när dess tidsintervall slår till för helger och vardagar.

Det används en PIR - sensor som med hjälp av infrarött ljus kommer att aktivera larmet ifall något rör sig. Då startar även en buzzer som ger en stark och hög larmsignal för att skrämma och varna de som gjort inbrott att larmet har gått och förhoppningsvis kommer assistans till platsen snabbt.

Ett exempel på användning är när man till exempel låser sitt hem eller kontorsbyggnader.

Design

Ett larm måste självklart ha olika komponenter som samarbetar ihop med varandra. Larmet är kopplat väldigt kompakt på en breadboard där alla komponenterna får plats. Hjärnan i arbetet är en Maple mini/Bluepill/STM32 som innehåller en MCU som sköter de olika instruktionerna i våran kod.

Komponenter i vårt larm:

- PIR - sensor - Infraröd sensor som läser in till programmet ifall den känner av något som rör sig/fått avslag.
- MCU - Bluepill/maple mini/STM32 - Mikrokontroller som gör instruktionerna.
- Buzzer - Utför ljudet som kommer från larmet ifall PIR - sensorn får avslag.
- Realtidsklocka DS3231 RTC - Klocka som går i realtid.
- Pull Up resistorer på 10kohm - fördelar spänningarna rätt för våran keypad.
- Kablar som sammankopplar hela systemet.

Alla dessa komponenter har olika prioriteter och alla komponenter är väsentliga för att larmet ska fungera som vi vill. Larmets konstruktion och uppkoppling finns inuti fritzing kretsschemat.

Programdesign

Larmet är byggd utav fyra stycken olika tasks och en semaphore. Larmet har inga interrupts, timers eller köer. I vårt larm behövs det inte några interrupts, köer eller timers. Den går enbart efter en semaphore som tilldelas om den ej är upptagen och sedan körs hela programmet efter man använt sin vTaskStartScheduler(); funktion. Vilket är ett måste för att ens kunna starta våra tasks. Våran realtidsklocka och keypad är de som är våra villkor för larmets funktionalitet.

Vi använder olika bibliotek som är obligatoriska för att programmet ska fungera. De flesta biblioteken är redan klara.

Kodens exakta funktionalitet och exakta konstruktion är att först väljer vi de bibliotek vi behöver för komponenterna som tex. uRTCLib.h biblioteket för realtidsklockan DS3231 o.s.v. Sedan deklarerar vi och gör några defines för de portar vi har använt som på STM32 är annorlunda jämfört med andra MCU:er som inte klarar av freeRTOS funktionalitet.

Vi har gjort ett eget **objekt** i programmet där vi sparar våran tid vad den är på dygnet och om det är helgtid nu eller vardag.

```
struct LarmSchedule {
  int StartTime;
  int EndTime;
} WeekSchedule[2] = {
  {480, 1020},
  {0, 1440}
};
```

Vi har även två till objekt som tar in data från WeekSchedule[ ] arrayen som får kollar ifall RTC tidsintervallet är helg - eller vardagstid. De objekten ser ut såhär:

```
 struct LarmSchedule *Workday = (struct LarmSchedule *)&WeekSchedule[0];
 struct LarmSchedule *Weekend = (struct LarmSchedule *)&WeekSchedule[1];
```

Våra tasks:

xTaskCreate(ClockTask = Task för våran RTC(realtidsklocka).

xTaskCreate(BuzzerTask = Task för våran buzzer/alarm signal.

xTaskCreate(SensorTask = Task för våran PIR - sensor.

xTaskCreate(KeypadTask = Task för våran keypad vi slår in våran pin kod med för deaktivering och aktivering utan att realtidsklockan stör.

xTaskCreate(ClockTask:

I denna task kropp så definieras realtidsklockan i en oändlig loop. Där refreshar vi och skriver ut den reella tiden så den alltid är rätt men även här så har vi en semaphore som vi måste bli tilldelad innan klockan får användas. Vi skriver ut realtiden i serial monitor ifall vi har fått tillgång till den av semaphoren. Får vi den inte så gör vi en delay och sedan testar igen.

xTaskCreate(BuzzerTask:

I den här task kroppen så loopar vi en läsning av en PIR - sensor som kollar ifall den är en 1:a från sin input signal, och då så kommer ett larm att gå så länge om klockan är i sitt tidsintervall för att larmet är aktiverat eller ifall vi själva manuellt har aktiverat larmet. Så länge larmet är avstängt så läser man inte ens från sensorn.

Så länge som man kan läsa en etta från PIR - sensorn så kommer även att en buzzer att låta med hög signal då. Buzzern i denna task aktiveras enbart ifall man får ut en 1:a ifrån våran PIR- sensor.

xTaskCreate(SensorTask:

I den här tasken så har vi alla villkor för om att vårt larm är eller kommer bli aktiverat och avaktiverat. Antingen genom våra tidsintervall från våran RTC eller om vi manuellt har hanterat dess aktivering och avaktivering. Allting i denna task är beroende på vad vi har gett dem för prioritet. Och denna task är beroende utav att vi fått våran semaphore också.

Mellan klockan 08.00 och 17.00 kommer våran RTC själv avaktivera larmet på vardagarna den tiden Mån - Fre. Utöver den tiden på vardagarna kommer larmet vara aktiverat. Helgerna är larmet helt på från 17.00 på Fredagen till 08.00 på Måndagen då ingen är på kontoret. Men är de nu så att man ska jobba helg över så får man avaktivera larmet innan man går in med en kod. Men då måste man aktivera larmet igen med samma kod.

xTaskCreate(KeypadTask:

I den här tasken så har vi även här lagt in mer villkor utöver våran realtidsklocka. Som vi beskrev förut så jobbar larmet automatiskt mellan olika tidsintervall med hjälp utav RTC:n men den är delvis även manuell. Här i denna task så definierar vi den manuella pin kod som kan aktivera och avaktivera larmet. I tasken så har vi olika arrays med bestämt pin lösen kollar av ifall de du trycker på vår keypad överensstämmer med pin koden som vi har satt.

Lampor/leds som även indikerar ifall du har tryckt på en utav knapp-inputsen och fått ett avslag. Skriver du rätt så startar du larmet ifall det är av och stänger av larmet ifall det är på. Då keypaden inte alltid ger avslag från knapptryckningarna är indikeringslampor bra.

I denna task blir manuella bool variabler som är deklarerade som “false” att bli “true” och tvärtom ifall man skrivit in rätt kod olika tider på dygnena.

Erfarenheter

När man gör ett projekt får man alltid nya erfarenheter. Vi som grupp har lärt oss jobba tillsammans och kunnat slutföra ihop ett projekt. Det är viktigt att det finns kemi i gruppen för att allt ska gå bra. Vårt utvecklingsarbete har varit från början tillsammans. Vi arbetade som grupp för att slutföra ett program genom att alla tog del i utveckling av programmet.

Parprogrammering var nyckeln till att alla kunde ta del i utvecklingen av programmet som vart slutligen ett i princip ett komplett larm med vissa finjusteringar som lämnades till sina öden pga tidsbristen på slutet av projektet.

Vi delade i slutet upp arbetet med presentationer, dokumentation och kod kommenteringar till olika personer i gruppen. Det viktigaste var att alla fick ta del av utvecklingen av programmet så alla förstått vad våran programkod gör och hur det kan användas.

Vi ödsla mycket tid på olika saker som tog tid eller inte alls gick att implementera. Som till exempel våran Keypad inte hade bibliotek som fungerade till våran MCU och fick lägga ner mycket tid på det. Att även dess knappar inte alltid gav avslag så fick vi även lägga till olika leds för att indikera ett avslag på knapparna ifall man skrivit fel kod eller inte.

Vi hade tänkt oss använda oss utav en LCD display med men tiden fanns ej för det då projektet skulle vara färdigt i tid. Även hade vi planer på att löda ihop allting med på ett kretskort men den tiden kunde vi drömma om att få.

Andra saker är kompilations fel, MCU kraschar och inga port avläsningar som gjorde att tiden bara blev mindre och mindre att slutföra arbetet.

Men framgångar har vi haft mest genom att vi fick koden till slut att göra som vi ville och att inga andra komponenter hade fel på sig.

Råd till andra som ska göra liknande system som vi har gjort är att ha en plan för vad ni vill göra och tänk så smått ni bara kan som fungerar innan ni hoppar på något nytt. Har ni något som fungerar så bygg vidare på det innan ni bygger vidare på något, som ska till något som inte fungerar ännu.

Slutsats

Ett roligt projekt att göra med en bra grupp är det som krävs för att något ska lyckas. Börjar man smått och utvecklar vartefter man får saker att fungera så kommer man komma längre än man tror.