Dokumentasjon/Beskrivelse av virkemåte:

Pseudo oppkobling:

SR04-sensoren skal sende ut trigger-signal og lytte på ekko for å gjøre en avstandsmåling for inngangen alarmen skal «vokte». Her setter jeg avstanden til å være 150cm og jeg ønsker å varsle ved å gi strøm til en buzzer som igjen vil gi fra seg en alarmerende lyd, dersom det er noen eller noe som «forstyrrer» lydsignalene.

Jeg setter det også opp slik at en RGB-LED-katode vil lyse rødt dersom enheten alarmerer, og grønt dersom alt er greit.

Det er også koblet opp en DS3231 i2c RTC modul, som vil kunne gi real time-utlesing av tid til Arduinoen. Denne har også et batteri for å kunne holde styr på klokken (opptil 8 år med riktig batteri ifølge forelesning 8), selv om Arduinoen er koblet fra en strømkilde, på samme måte som et BIOS-batteri i en PC, eller batterier i gamle GBC-spill.

Live klokkeslett skal vises på TFT-skjermen og skjermen skal lese ut «OK» i grønt med klokkeslett foran dersom alarmen ikke er utløst, og «ALERT» i rødt med klokkeslett foran dersom noe forstyrrer avstanden og den blir lavere enn 150cm.

Komponenter brukt:

-1x Arduino (Det er brukt en Arduino UNO rev. 3)

-1x Adafruit 1.14" 240x135 Color TFT Display

-1x Generic buzzer

-1x Breadboard

-Div. hann-til-hann-kabler

-1x RGB-LED katode

-3x 220 Ohm motstand

-1x HC-SR04 ultrasonisk sensor

-1x DS3231 I2C RTC modul

Oppkobling:

Kobling av TFT-skjerm

Vin er koblet til 5V på Arduino via breadboard.

GND er koblet til jord på Arduino via breadboard

SCK er koblet til digital pin 13 på Arduino via breadboard

MISO er koblet til digital pin 12 på Arduino via breadboard

MOSI er koblet til digital pin 11 på Arduino via breadboard

TFTCS er koblet til digital pin 10 på Arduino via breadboard

DC er koblet til digital pin 8 på Arduino via breadboard

SOCS er koblet til digital pin 4 på Arduino via breadboard

Kobling av HC-SR04

VCC er koblet til 5V på Arduino via breadboard

GND er koblet til jord på Arduino via breadboard

TRIG er koblet til digital pin 3 på Arduino via breadboard

ECHO er koblet til digital pin 2 på Arduino via breadboard

Kobling av buzzer

PWP er koblet til Analog port 0(A0) på Arduino via breadboard

GND er koblet til jord på Arduino via breadboard

Kobling av RGB-LED katode

R(rød) er koblet til en 220ohm motstand på breadboard for så å gå videre til digital pin 5 på Arduino

G(grønn) er koblet til en 220ohm motstand på breadboard for så å gå videre til digital pin 6 på Arduino

B(blå) er koblet til en 220ohm motstand på breadboard for så å gå videre til digital pin 7 på Arduino

Gnd er koblet til jord på Arduino via breadboard

Kobling av DS3231

GND er koblet til jord på Arduino via breadboard

VCC er koblet til 5V på Arduino via breadboard

SDA er koblet til analog input 4(A4) på Arduino via breadboard

SCL er koblet til analog input 5(A5) på Arduino via breadboard

CR2025 batteri er koblet til enheten

Virkemåten slik jeg forstår den:

Det er en SR04 ultrasonisk sensor som sender ut og mottar ultrasoniske lydsignaler (trigger og echo) og man kan bruke den verdien man leser fra sensoren med en matematisk konvertering for å måle en avstand. Denne sensoren har en relativt kort rekkevidde, men er relativt nøyaktig for avstanden som er brukt i dette prosjektet (opptil 150cm).

Denne målingen er ikke helt samtid da det går noen ms hver vei og lyden beveger seg i lydens hastighet, og ikke like rask som f.eks. Lysets hastighet. I dette prosjektet mener jeg fortsatt at denne vil fungere helt greit.

TFT-skjermen fra Adafruit er koblet opp med :

-5v inn

-jording

-SCK som er klokke-input signalet

-MISO er koblet til, men ikke i bruk, da den kun er relevant for å lese fra SD-kort

-MOSI som sender signaler fra Arduino, til mikrokontrolleren på kretskortet til TFT-skjermen som igjen sender bilde til skjerm.

-TFTCS er en TFT SPI cip selector pin

-DC er TFT SPI data/command pin velgeren.

-SDCS brukes ikke, men er en velger for SD-kort. Den bruket dersom man ønsker å lese fra SD-kort.

Denne skjermen er relativt fargesterk med grei «viewing angle» og oppløsning med tanke på størrelsen (260ppi). Apple watch 7(41mm) har til sammenligning 326ppi.

Skjermen sin jobb i dette prosjektet vil være å vise klokkeslett og alarmstatus.

RGB-LED katoden er koblet opp med en 220ohm motstand for hver av fargepolene og en jording på den negative polen. Her sendes da strøm til den polen som gir ønsket farge eller fler samtidig, for å skape en RGB-effekt. Dette vil da være verdier fra 0-255 i et typisk RGB-format eks. 255,255,255 for «hvitt» lys. Dette da selvfølgelig styrt i koden.

Buzzeren sin jobb er i dette prosjektet ene og alene å være irriterende, dersom alarmen går av. Den er særdeles primitiv og er koblet opp mot jord, og har en digital pin som går fra Arduino og til en 100ohm motstand, for så å gå inn i på den positive polen til buzzeren.

Dette gjør at den er enkel å utløse ved å gi den spenning ved ett enkelt kall i koden, som kan kjøres i det lydsignalet til den ultrasoniske sensoren blir forstyrret.

DS3231 modulen sin jobb er i dette prosjektet å holde styr på klokkeslett. Den har også ekstra funksjonalitet som ikke blir brukt nå, nemlig en EPROM: Erasable programmable read-only memory.

Dette er minne man kan skrive til som ikke er volatilt, som vil si at det ikke trenger konstant spenning for å holde noe lagret på seg. Dette er minne man kan skrive til, for å lagre små mengder data, som raskt kan leses av. Som om ikke dette var nok, så har den en innebygget temperatursensor og plass til et litium-batteri.

Denne funksjonaliteten brukes da ikke i dette tilfellet. Det som brukes her er klokke-funksjonaliteten, nærmere bestemt RTC. Real Time Clock som også er veldig nøyaktig og hardfør.

Denne modulen bruker en I2C Serial bus og har en Gnd til jord og en Vin til 5V koblet til.

Den har også en SCL pin som går til I2C klokkepinnen på Arduino UNO (A5) for frekvens og en SDA pinne som går til I2C data pinnen på Arduino UNO (A4) for å kunne overføre data fra DS3231-modulen og over til Arduinoen(slave og master). For at dette skal fungere jeg i dette tilfellet helt avhengig av bibliotek for Wire.h og RtcDS3231.h.

I2C er gunstig for lav hastighet applikasjoner som dette, men vil være mindre gunstige i ting hvor lav forsinkelse og høy oppdateringshastighet/frekvens er av høyeste prioritet.

Etter hva jeg har forstått så vil dette på en veldig forenklet måte fungere slik:

En start-condition

SDA linken skifter fra høy til lav spenning før SCL linken skifter fra høy til lav spenning og det motsatte vil skje i stop-betingelse.

Mellom start og stop-betingelsen vil man sende «data-frames» på 8-bit med informasjon hvor MSB er først og LSB sist. Mellom hver «frame» er det en ACK/NACK som verifiserer at en «frame» har blitt mottatt korrekt. Hver ACK-bit skal være mottatt hos mottaker, før neste «dataframe» blir sendt. Etter at alle «frames» har blitt sendt, vil da som tidligere forklart «master» i dette tilfellet sende en stop-betingelse til «slave», for å varsle at overføringen er over og SCL linken vil gå til høy, fra lav påfulgt av et skifte fra lav til høy i SDA linken.

Referanser til data-ark og forklaringer:

I2C:

https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

TFT-skjerm:

https://learn.adafruit.com/adafruit-1-14-240x135-color-tft-breakout

RGB-LED:

https://create.arduino.cc/projecthub/muhammad-aqib/arduino-rgb-led-tutorial-fc003e

HC-SR04:

https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/