Teknisk rapport

Fused

Inger Marie Liepelt

Rifat Naim Islam

Gorm Osborg

Marianne Ludviksen Winge

Frida Rønning

10.06.20

IN1060 Vår 2020

Universitetet i Oslo

1. Mål for prosjektet

Problemstillingen vi ville løse var hvordan oppfylle sosiale behov ved å tilrettelegge for lavterskel kommunikasjon nå som COVID-19 har endret måten vi møtes på. Prototypen vi har laget skal løse dette problemet ved at brukeren har mulighet til å stille seg selv tilgjengelig ved å slå på sin blomst og dermed signalisere at de er klare for en prat, og at andre kan stille seg tilgjengelig for brukeren ved å gjøre det samme. Dette er en lavterskel måte å kommunisere sosialt behov på, fordi alt de må gjøre er å trykke på knappen for å signalisere tilgjengelighet.

2. Kort presentasjon av hva videoen viser

Videoen viser først et scenario for å vise brukskontekst og hvordan det er tenkt at blomsten skal svare på problemstillingen.

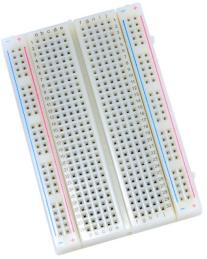
Deretter følger en introduksjonsvideo som viser blomstens funksjoner og hvordan vi bruker en FTP-server for å sjekke status på andre blomster: tilgjengelig (1) eller ikke tilgjengelig (0). Vi har kun laget en blomst, så for å simulere statusen til andre blomster, oppdaterer man de korresponderende filene på serveren manuelt for å vise om en venn har "slått på" eller "slått av" sin blomst. Toveis kommunikasjon med flere andre blomstene via HTTPS (GET) og FTP (POST) er altså implementert i koden, men kopier av den fysiske prototypen har ikke blitt laget av ressursbesparende hensyn.

3. Link til videoen

LINK HER

4. Dokumentasjon av den tekniske løsningen i detalj:

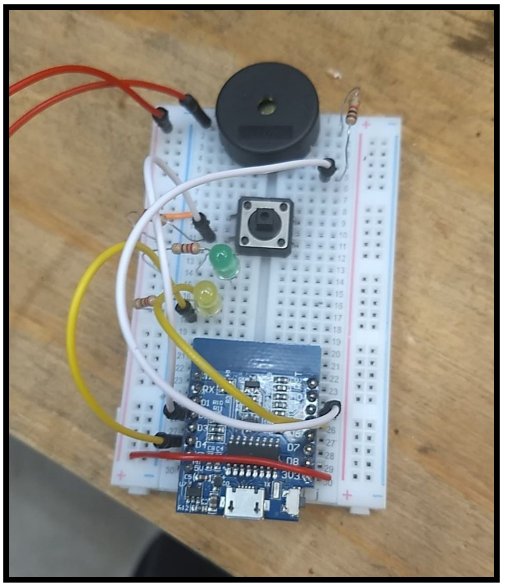
Komponenter	Antall	Type	Bilde
Mikrokontroller	1	Wemos D1 Mini	
Strømforsyning	1	5V DC, USB->Micro USB (mobillader)	

PCB	1	Dobbeltsidig Prototype PCB, 5x7cm	
Breadboard	1	Breadboard, 14x10cm	
Resistor	4	3x 1KΩ, 1x 10KΩ	
Knapp	1	Illuminated pushbutton 22.3mm green 24v	
LED-diode	6	Gul LED-diode 3v	
Lydelement	1	Piezoelektrisk forsterker	
Kobbertråd	30cm	1.5 mm hel kobbertråd	
Koblingsledning	6x 20cm	1mm, tvinnet kobbertråd	

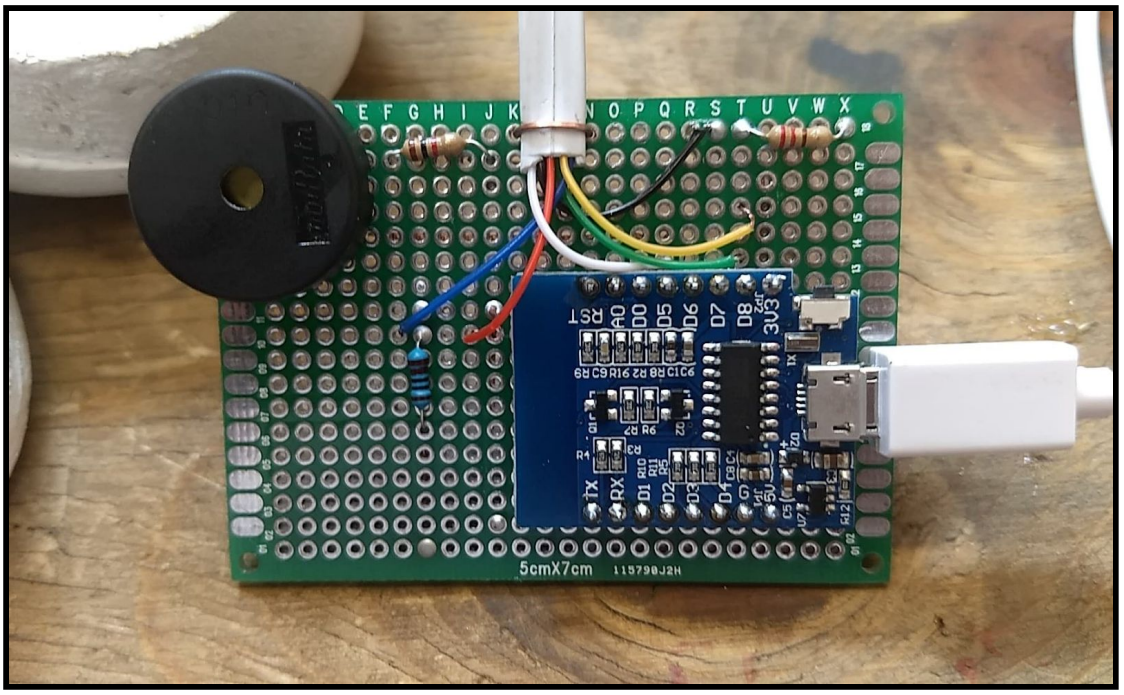
Loddemetall	20cm	2mm loddetinn	
Loddefett	50g	Felder Loddefett	
Loddebolt	1	Loddebolt m/ tilbehør og holder, 350C	

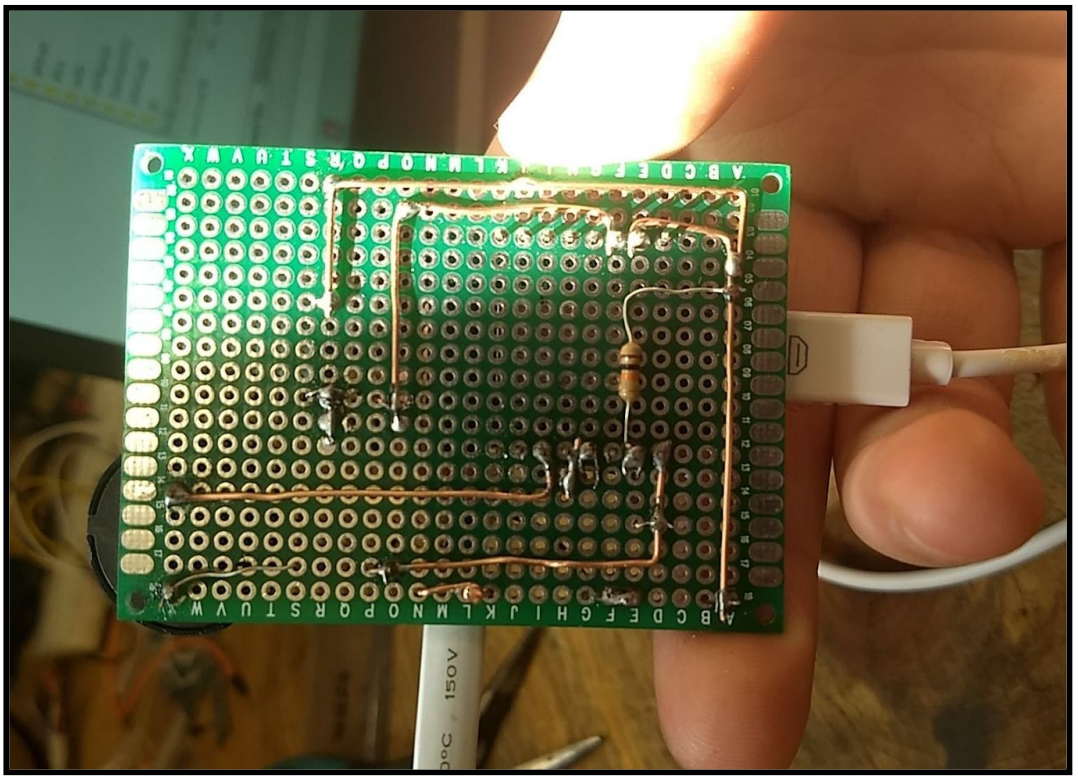
Kretsen ble først bygget på et breadboard, koblet til en Arduino Uno og en ESP8266-modul (Wi-Fi modul), hvor logikksignalet ble routet gjennom en 5v-3.3v toveis logisk nivåomformer. Det viste seg å være svært krevende å kommunisere mellom Arduino Uno og ESP8266, så Arduino/ESP8266-oppsettet ble byttet ut med en Wemos D1 Mini, som er en mikrokontroller med integrert ESP8266.

Wemos D1 Mini kan kobles opp til USB-port på PC via en standard "USB-til-microUSB"-kabel for strømforsyning og bootloading. For PC-uavhengig strømforsyning trenger man kun å koble på en 5V DC adapter, bedre kjent som "mobillader".

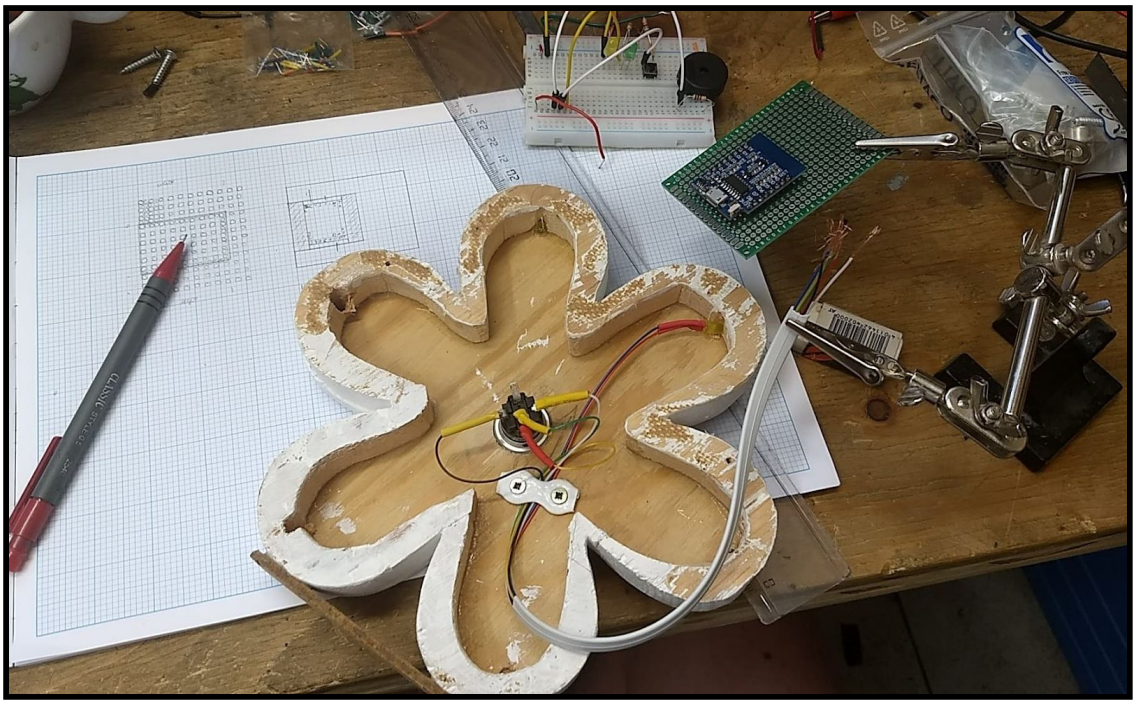


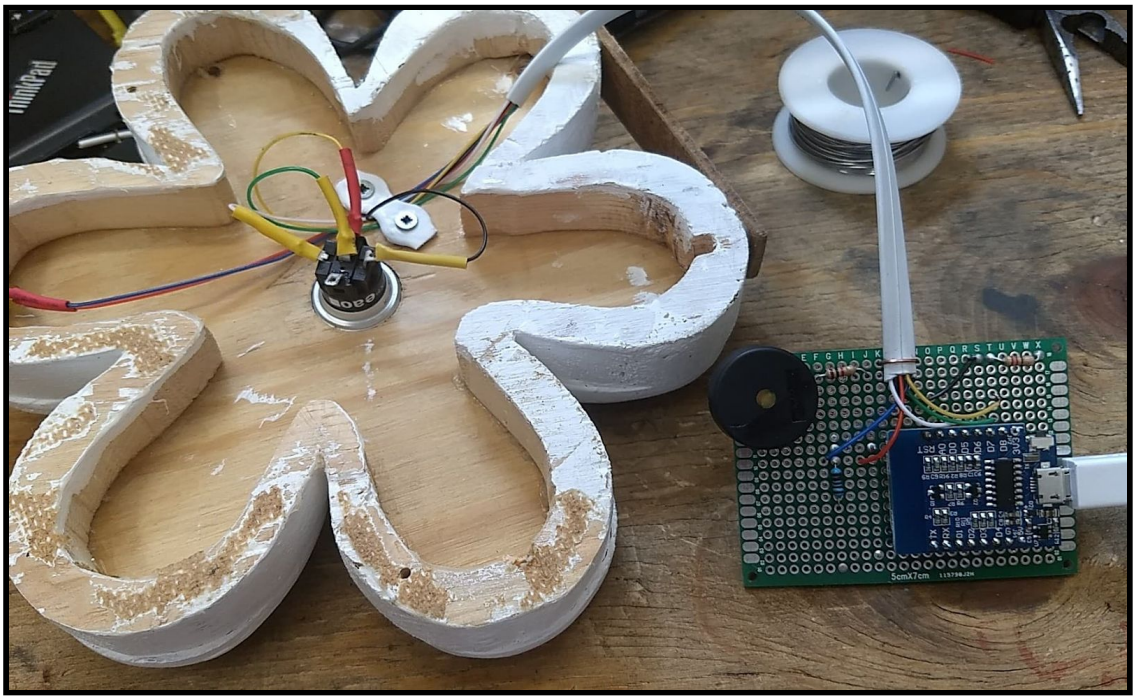
Når kretsen fungerte på breadboardet ble den tegnet opp på papir. Deretter ble kretsen loddet på et dobbeltsidig PCB prototypebrett, hvor 2mm kobberkabler ble brukt som strømledere. Piezoen er loddet rett på prototypebrettet.





En kabel med 6 isolerte mindre kabler ble loddet på prototypebrettet og trukket ut til baksiden av blomsten. 4 av disse ble loddet på den bakgrunnsbelyste knappen, og 2 loddet på en gul LED-diode. Både knappen og LED-dioden var allerede limt fast i blomsten. De 5 LED-diodene som er i de andre 5 kronbladene er ikke tilkoblet kretsen, for enkelhets skyld, og er alltid "døde". Disse kan bli koblet på ved senere anledning, hvis det trengs for enda mer høyoppløselig prototyping.





For at blomsten skal virke fullstendig trenger den kun strøm via strømforsyning, samt et tilgjengelig trådløst internett den kan koble seg til.

5. Forklaring av kode

Github-link

Når brukeren trykker på knappen i midten av blomsten skrur bakgrunnslyset seg på for å signalisere at man selv er i "tilgjengelig"-modus. Samtidig opprettes en tekstfil, statusX.txt (hvor X er ID til din blomst), som inneholder en streng "1" for "tilgjengelig". Tekstfilen sendes til en FTP-server på

osborg.no/flower/status1.txt (eller overskriver denne filen om den allerede eksisterer) via WiFi. Metodene wiFiSetup(), ownStatus() og getOtherStatus() tar hånd om internettilkobling og fildeling, de to siste via doFTP() og eRcv(). SPIFFS blir brukt for filhåndtering. Når knappen trykkes inn igjen, skrur bakgrunnslyset seg av for å signalisere at man er i "utilgjengelig"-modus. Da oppdateres osborg.no/flower/status1.txt til å si "0" for "utilgjengelig".

Blomsten sjekker f.eks. https://www.osborg.no/flower/status1.txt (eller andre statusfiler) for status-filer til andre blomster hvert 30 sekund via en HTTPS GET-request. Hvis den finner at andre blomster er tilgjengelige registreres dette på arduino lokalt, og det korresponderende kronbladet lyser opp. For eksempel, hvis Ola har Blomst nr. 5, og status5.txt sier "1", lyser Ola sitt kronblad opp.
Så lenge din egen blomst er i tilgjengelig-modus, spilles det også et kort lydvarsel når andre blomster blir tilgjengelige.

6. Forklaring av bibliotekene

De følgende bibliotekene har blitt brukt:

- **<ESP8266Wifi.h>**: WiFi-spesifikke rutiner som lar ESP8266-modulen på mikrokontrolleren koble til internett.
- **<ESP8266HTTPClient.h>**: HTTP-spesifikke rutiner som lar mikrokontrolleren opprette klienter som kan gjøre GET-requests til internettsider. Lar deg hente informasjon fra nettsider.
- **<WiFiClientSecureBearSSL.h>**: HTTPS-spesifikke rutiner som lar mikrokontrolleren lage sertifikater og fingerprints så man kan bruke HTTPS-protokoll. Nødvendig for å gjøre GET-requests til sikre nettsider (HTTP Secure)
- **<FS.h>** : SPIFFS-spesifikke rutiner som utvider mikrokontrolleren sitt filsystem så filer kan sendes og mottas via FTP, og for å kunne lese og skrive til fil.
- **<SensorToButton.h>**: Lar bruker opprette objekter av knappe-klassen som kommer med integrert debounce-funksjonalitet.