NO_CAPTION Macintosh HD:Users:LindaEllinore:Downloads:Malmostad_PAGY_liggande.jpg

# Gymnasiearbete Smartwatch

*Att skapa en billigare smartwatch på den ESP32-baserade M5Stack Core2-plattformen*

Gymnasiearbete

TE18TE

Teknikprogrammet

Pauliskolan

2020/21

Hugo Strömberg

Niklas Ku

Handledare:

Axel Månsson

## Abstract

## Innehållsförteckning

[**Gymnasiearbete Smartwatch**](#_9a2ve368g58w) **0**

[**Abstract**](#_r26rvr1cxuh7) **1**

[**Innehållsförteckning**](#_is0fn1odyykh) **2**

[**1. Inledning**](#_ivfl0pnlke74) **4**

[1.1 Bakgrund](#_ytfhgtsquxk7) 4

[1.1.1 Teoretisk Bakgrund](#_850ys7uwcl4p) 4

[1.2 Syfte](#_3yghn5ko40cn) 6

[1.2.1 Specifikationer](#_w1efe2w3f55y) 6

[1.2.2 Avgränsningar](#_d2u9gta01ui6) 7

[1.3 Material](#_x3cz0wrngj77) 8

[1.3.1 Hårdvara](#_3zvpwm9szn56) 8

[1.3.2 Program och mjukvara](#_6acivuym8d1h) 8

[1.4 Metod](#_typk96i3ajrk) 10

[1.4.1 UIFlow och drivrutiner](#_hlhg8oe25y5k) 10

[1.4.2 Internetuppdaterad digitaltid](#_kilzquwl6pzu) 11

[1.4.3 Tidtagare och timer](#_i39a1qf602bv) 12

[1.4.3.1 Tidtagare](#_iorxgb5jdg68) 12

[1.4.3.2 Timer](#_39bdhsbspb5h) 14

[1.4.4 Batterimätning](#_hs7wxx79b0n1) 14

[1.4.5 Google Calendar](#_o2di0yd8sr33) 14

[1.4.5.1 Mosquitto MQTT](#_w7f9f86jwacq) 14

[1.4.5.1.1. Installation av Mosquitto](#_mu0bvui0gye1) 16

[1.4.5.1.2 Test av Mosquitto](#_gom2jl62xyzw) 16

[1.4.5.2 Node.js](#_c9cd3657pv5y) 16

[1.4.5.2.1 Node Red](#_y6vb1xpxxwq) 16

[1.4.5.2.2 Google Cloud Platform](#_686jqlt43bgq) 16

[1.4.5.3 Överföring till Core 2](#_sd18aqaa6vyo) 16

[1.4.6 Banddesign och hållare](#_nc91hljtfyyb) 17

[**2. Resultat**](#_1uwljxjqd3tk) **18**

[2.1 Interna funktioner](#_6cd2ztjmiu0e) 18

[2.2 Internetkopplade funktioner](#_hfmpwitlsloa) 19

[2.3 Bandet och hållare](#_yd18pmlu4pjm) 20

[**3. Diskussion**](#_fm3d9qajrqrm) **21**

[3.1 Problem](#_30yhiaydjzjr) 21

[3.2 Erfarenheter](#_sjm62i4eakza) 22

[3.3 Slutsatser](#_zbs4p3uf2zpz) 23

[**4. Källförteckning**](#_4gsn3948yk38) **24**

[**Bilagor**](#_xx8je6wr4ezq) **25**

[***Projektplan***](#_isillnc8o61l) ***27***

[***Innehållsförteckning***](#_eo58gbkww1ma) ***28***

[***1 Bakgrund***](#_9072421wd4z6) ***29***

[***2 Syfte***](#_fmuuy9fo6bxc) ***30***

[*2.1 Specifikation*](#_bx6h3vhvrpre) *30*

[*2.2 Experiment*](#_twv52a4s1x2w) *31*

[*2.3 Resultat*](#_gr1t292a9v10) *31*

[***3 Metod***](#_jgde1arg9qfe) ***32***

[*3.1 Hårdvara*](#_36ubxvm7wsfb) *32*

[*3.2 Mjukvara*](#_sfegijg6d1v) *32*

[***4. Bilagor***](#_7laumlsxfvlo) ***33***

## 

## 1. Inledning

### 1.1 Bakgrund

Smartklockor har funnits i olika former sen sent 70-tal men har först på senare år tillgängliggjorts för en bredare användarbas. Författarna är två teknikelever på Pauliskolan i Malmö som båda funnit att smartklockor fortfarande är för kostsamma för gemene man. Under de senaste åren har starkare mikrokontroller än någonsin tidigare släppts. En av dessa är ESP32 som är tillräckligt både liten och strömsnål för att fungera som processor till en smartklocka. Därför valdes projektet att skapa en billigare klocka av tillgängliga komponenter.

#### 1.1.1 Teoretisk Bakgrund

**Mikrokontroller:** Liten dator - CPU, arbetsminne och programminne, samt ofta I/O integrerat på en och samma bricka.

**ESP32:** Billig mikrokontroller med både wi-fi och bluetooth.[[1]](#footnote-0)

**M5Stack Core 2:** Plattform som integrerar en ESP32 med ett batteri, skärm, knappar och ett flertal andra funktioner för enkel användning till utveckling av IOT produkter.[[2]](#footnote-1)

**M5Stack UIFlow:** Grafiskt IDE för utveckling på M5Stacks olika produkter.[[3]](#footnote-2)

**IDE:** Integrated Development Environment - program för att skriva program.

**M5Stack Burner:** Mjukvara för att bränna drivers till M5Stacks produkter.[[4]](#footnote-3)

**Firmware:** Mjukvara inprogrammerad i hårdvaran.

**COM Port:** Kommunikationsport.

**SSID:** Service Set IDentifier - Standard för nätverksnamn.

**MQTT:** Nätverksprotokoll och kommunikationsprotokoll som fungerar med tre parter. Meddelanden publiceras till en **server** av en **klient**, varefter meddelandet skickas vidare till alla **prenumererade** klienter. MQTT är ett binärt protokoll till skillnad från http som är ett ASCII-baserat protokoll. Detta innebär att MQTT är mindre nätverkskrävande.

**Mosquitto:** Öppet verktyg för att skapa egna MQTT servrar.[[5]](#footnote-4)

**7-Zip:** Öppet program för hantering av komprimerade filer. [[6]](#footnote-5)

**JavaScript:** Programmeringsspråk som vanligtvis används i webbläsare för att låta dem exekvera mer avancerade funktioner.

**Node.js:** Öppet back-end exekveringsmiljö för javascript genom v8-motorn. Används inom servrar för att exekvera javascript utanför en webbläsare.[[7]](#footnote-6)

**Node Red:** Flödesbaserat verktyg för visuell programmering av Node.js i en webbläsare.[[8]](#footnote-7)

**Google Cloud Platform:** Googles plattform för att tillgängliggöra sina tjänster till slutanvändare. Genom denna får vi tillgång till kalendern och informationen däri.[[9]](#footnote-8)

**RTC:** Står för Real-time clock alltså realtidsklocka som bevakar den aktuella tiden.[[10]](#footnote-9)

### 

### 1.2 Syfte

Projektets mål är att bygga en enkel och billig klocka av enkla komponenter som gemene man kan tillhandahålla. Klockan ska till den grad det går ha tillhandahålla samma funktioner som konkurrerande klockor.

#### 1.2.1 Specifikationer

Klockan ska kunna:

* Visa tiden analog och/eller digital klocka
* Ta tiden eller gör en nedräkning
* Visa tid och batteriprocent i skärmens topp
* Visa datum och visa händelser med koppling till google-kalender
* Ta emot notiser från telefon via bluetooth samt signalera ägaren med vibration eller ljud
* Ta emot samtal och ringa genom kontaktlista eller digitalt numeriskt tangentbord genom mobilen
* Visa väder
* Visa både klockans och en kopplad telefons batterinivå
* Kontrollera musikspelare på telefonen
* Spela något enklare spel, förslagsvis tetris (extra, om vi hinner)
* Visa puls samt spara värdena
* Visa de olika funktionerna på varsin “panel” eller sida som är åtkomlig genom genvägar på en huvudmeny
* Justera “Stör ej” funktion för både smartklockan och mobilen
* Spara batteri med anpassningsbar tidsfunktion genom en skärmsläckande funktion
* Justera mellan dag- och nattläge
* Ge användaren enklare navigering med bilder på knappar istället för text
* Välja bluetoothuppkoppling
* Byta tid
* Justera ljusstyrka och volym

#### 1.2.2 Avgränsningar

Projektet har på handledarens inrådan begränsats genom att en Core 2 införskaffades istället för att en del av projektet skulle vara att samla delar till och bygga själva klockan. Fokus hamnar då istället på mjukvaran samt att designa ett armband och hållare till Core 2 som gör den bärbar.

Detta valdes för att tillåta mer fokus och tid till experimenten och programmeringen.

### 

### 1.3 Material

#### 1.3.1 Hårdvara

|  |  |  |  |
| --- | --- | --- | --- |
| Produkt | Syfte | Pris | Länk |
| M5Stack Core 2 | Utvecklingsmodul med passande komponenter till en smartklocka. | 1st 456,25 kr  2st 912,5 kr | [Länk](https://www.elfa.se/sv/esp32-m5core2-utvecklingsmodul-m5stack-k010/p/30181494?queryFromSuggest=true)[[11]](#footnote-10) |
| Pulssensor | Komponent som används till att mäta/ ta pulsen. | 1st 288,75 kr  2st 577,5 kr | [Länk](https://www.elfa.se/sv/pulssensor-sparkfun-electronics-sen-11574/p/30152838?q=puls+sesnor&pos=6&origPos=6&origPageSize=10&track=true)[[12]](#footnote-11) |
| 510 mAh 3.7V Li-Po batteri | Används istället för M5Stack Core 2:s eget batteri. Byts ut för längre livstid. | 1st 220 kr  2st 440 kr | [Länk](https://www.elfa.se/sv/laddningsbart-batteripaket-li-po-7v-510mah-renata-icp303450pa-02/p/30158706?q=*&pos=1&origPos=2&origPageSize=10&track=true)[[13]](#footnote-12) |
| Onyx och/eller PLA filament till 3D-printern | Används för att 3D-printa Fusion360 modellen av klockans hållare och band | Gratis, då skolan erbjuder gratis användning av 3D-printern och filament | [Länk till Onyx](https://markforged.com/product/onyx-filament/)[[14]](#footnote-13)  PLA filament länk |

Utöver ovanstående krävdes för genomförande av projektet tillgång till en dator för kodskrivning och programmering samt överföring till Core 2. Vidare behövs en 3D-skrivare för att utföra en utskrivning av hållaren och bandet.

#### 1.3.2 Program och mjukvara

* M5Stack UIFlow
* M5Burner
* Mosquitto MQTT
* 7-Zip
* Node.js
* Node Red

*(se* [*1.1.1 Teoretisk Bakgrund*](#_850ys7uwcl4p) *för länkar till hemsidor)*

All använd programvara var i skrivet tillfälle tillgänglig utan kostnad.

### 

### 1.4 Metod

#### 1.4.1 UIFlow och drivrutiner

Första steget är att kontrollera om Core 2 modulen och dess komponenter fungerar med hjälp av det förprogrammerade fabrikstest firmware.

För att använda mjukvaran UIFlow för att programmera Core 2 ska M5Burner, UIFlow-Desktop-IDE och CP2104-Driver laddas ner på datorn från [M5Stacks egen hemsida](https://m5stack.com/pages/download). I M5Burner letas “UIFlow\_Core2” upp med en grön “Official” markör. Välj senaste versionen och ladda ner den. Koppla in Core2 till datorn med den inkluderade USB-A till USB-C kabeln, välj rätt COM i M5 Burner och sätt baudrate till 1500000 sedan tryck “Burn”. Ett nytt fönster kommer dyka upp där SSID ska skrivas in samt lösenordet. Sedan trycks “Start”. COM-monitorn kommer dyka upp som följande:





Det ska stå “Burn Successfully” och Core 2 borde startas om. Om det står “Burn Failed” ska COM kontrolleras och pröva “Burn” igen eftersom det kan ibland misslyckas men lyckas därefter. Nästa steg är att trycka på “Flow” och sedan “USB” på Core 2-menyn, modulen kommer att startas om. Öppna upp UIFlow-Desktop-IDE på datorn, tryck tillåt om datorn frågar om IDEn och mottagandet av inkommande nätverksanslutningar, välj betaversionen sedan välj COM och Core 2-modulen och tryck ok. En grön rektangel ska dyka upp med texten “Connected”, om det är en röd rektangel borde COM kontrolleras och försök igen via “Refresh device status” vid botten av programmet. Kontrollera även att CP2104-Drivern är installerad genom att koppla in Core 2 till datorn, tryck på “Om den här datorn” på din Mac (längst upp till vänster - äpplet) → Systemrapport → USB, om “CP2104 USB to UART Bridge Controller” finns med är drivern nedladdat korrekt.

UIFlow använder programmeringsspråket micropython men är även kompatibel med python. När författarna programmerade mjukvaran bestämde de sig att varje funktion skulle programmeras i separata filer för att i slutet sätta ihop allt i ett. Detta gjordes för att bevaka processen enklare och kunna lättare hitta problem i koden.

#### 1.4.2 Internetuppdaterad digitaltid

Koden är inte komplex, den använder sig av internetuppkoppling för att hämta tid och datum. Detta läggs i setup men för att få den att uppdatera skapas en “label” under loop där tiden kan skrivas ut på skärmen.

|  |
| --- |
| wifiCfg.doConnect('SSID', 'PASSWORD') time = M5Label('Show time here', x=87, y=96, color=0x000, font=FONT\_MONT\_14, parent=None)  while True:  time.set\_text(str(rtc.printRTCtime()))  wait\_ms(2) |



wifiCfg står för WiFi Config vilket innebär att kunna konfigurera den trådlösa nätverket som ska användas.[[15]](#footnote-14) Med wifiCfg där SSID och lösenord ska skrivas in ska det försöka koppla upp sig till nätverket med doConnect vilket låter “RTC time” hämta den lokala tiden.

#### 1.4.3 Tidtagare och timer

##### 1.4.3.1 Tidtagare

I den slutliga versionen finns det med timmar och minuter som räknas upp men i test-filen används endast sekunder. I slutliga versionen krävs det endast att när sekunder blir 60 ska det bli 0 och minuter ska få +1, samma metod används för timmar när minuter blir 60. Tidtagaren har därför en gräns på 100 timmar.

|  |
| --- |
| start\_stop\_count = None resetcount = None sec = None seconds = M5Label('00', x=134, y=21, color=0x000, font=FONT\_MONT\_48, parent=None) start\_stop = M5Btn(text='start/stop', x=123, y=96, w=70, h=30, bg\_c=0xFFFFFF, text\_c=0x000000, font=FONT\_MONT\_14, parent=None) reset = M5Btn(text='reset', x=125, y=177, w=70, h=30, bg\_c=0xFFFFFF, text\_c=0x000000, font=FONT\_MONT\_14, parent=None)  from numbers import Number  def start\_stop\_pressed():  global start\_stop\_count, resetcount, sec  start\_stop\_count = (start\_stop\_count if isinstance(start\_stop\_count, Number) else 0) + 1  pass start\_stop.pressed(start\_stop\_pressed)  def reset\_pressed():  global start\_stop\_count, resetcount, sec  resetcount = (resetcount if isinstance(resetcount, Number) else 0) + 1  pass reset.pressed(reset\_pressed)  sec = 0 start\_stop\_count = 0 resetcount = 0 while True:  if start\_stop\_count % 2 == 1:  sec = (sec if isinstance(sec, Number) else 0) + 1  resetcount = 0  wait(1)  if resetcount % 2 == 1:  start\_stop\_count = 0  sec = 0  wait(1)  seconds.set\_text(str(sec))  wait\_ms(2) |





Start-knappen startar tidtagaren och stop-knappen stoppar tidtagaren. Reset-knappen återställer tidtagaren till 0 samtidigt som den stoppar tidtagaren. Detta görs med hjälp av variabel “counter” som räknar antal gånger en knapp har blivit nedtryckt. Varje gång start/stop är nedtryckt går räknaren upp +1, när värdet är udda börjar tidtagaren, när värdet är jämnt stoppas tidtagaren och nedtryckningen återställer även reset-räknaren. Reset-knappen har en egen räknare, när värdet är udda ska start/stop räknar återställas till 0 samtidigt som att tidtagaren återställs.

##### 1.4.3.2 Timer

Koden är mer komplex än tidtagaren, i timer eller tidur krävs det fler knappar och situationer gällande timmar och minuter.

|  |
| --- |
| count = None seconds = None hours = None minutes = None vibration\_count = None  h = M5Label('00', x=63, y=67, color=0x000, font=FONT\_MONT\_48, parent=None) m = M5Label('00', x=128, y=67, color=0x000, font=FONT\_MONT\_48, parent=None) s = M5Label('00', x=197, y=67, color=0x000, font=FONT\_MONT\_48, parent=None) h\_add = M5Btn(text='+', x=59, y=10, w=50, h=50, bg\_c=0xFFFFFF, text\_c=0x000000, font=FONT\_MONT\_14, parent=None) h\_reduce = M5Btn(text='-', x=63, y=111, w=50, h=50, bg\_c=0xFFFFFF, text\_c=0x000000, font=FONT\_MONT\_14, parent=None) m\_add = M5Btn(text='+', x=124, y=10, w=50, h=50, bg\_c=0xFFFFFF, text\_c=0x000000, font=FONT\_MONT\_14, parent=None) s\_add = M5Btn(text='+', x=197, y=10, w=50, h=50, bg\_c=0xFFFFFF, text\_c=0x000000, font=FONT\_MONT\_14, parent=None) m\_red = M5Btn(text='-', x=136, y=119, w=50, h=50, bg\_c=0xFFFFFF, text\_c=0x000000, font=FONT\_MONT\_14, parent=None) s\_red = M5Btn(text='-', x=213, y=119, w=50, h=50, bg\_c=0xFFFFFF, text\_c=0x000000, font=FONT\_MONT\_14, parent=None) start = M5Btn(text='start', x=130, y=186, w=70, h=30, bg\_c=0xFFFFFF, text\_c=0x000000, font=FONT\_MONT\_14, parent=None) touch\_button0 = M5Btn(text='Button', x=60, y=40, w=200, h=200, bg\_c=0xFFFFFF, text\_c=0x000000, font=FONT\_MONT\_14, parent=None)  from numbers import Number  def h\_add\_pressed():  global count, seconds, hours, minutes, vibration\_count  if hours >= 99:  hours = 0  else:  hours = (hours if isinstance(hours, Number) else 0) + 1  pass h\_add.pressed(h\_add\_pressed)  def m\_add\_pressed():  global count, seconds, hours, minutes, vibration\_count  if minutes >= 59:  minutes = 0  else:  minutes = (minutes if isinstance(minutes, Number) else 0) + 1  pass m\_add.pressed(m\_add\_pressed)  def s\_add\_pressed():  global count, seconds, hours, minutes, vibration\_count  if seconds >= 59:  seconds = 0  else:  seconds = (seconds if isinstance(seconds, Number) else 0) + 1  pass s\_add.pressed(s\_add\_pressed)  def h\_reduce\_pressed():  global count, seconds, hours, minutes, vibration\_count  if hours <= 0:  hours = 99  else:  hours = (hours if isinstance(hours, Number) else 0) + -1  pass h\_reduce.pressed(h\_reduce\_pressed)  def m\_red\_pressed():  global count, seconds, hours, minutes, vibration\_count  if minutes <= 0:  minutes = 59  else:  minutes = (minutes if isinstance(minutes, Number) else 0) + -1  pass m\_red.pressed(m\_red\_pressed)  def s\_red\_pressed():  global count, seconds, hours, minutes, vibration\_count  if seconds <= 0:  seconds = 59  else:  seconds = (seconds if isinstance(seconds, Number) else 0) + -1  pass s\_red.pressed(s\_red\_pressed)  def start\_pressed():  global count, seconds, hours, minutes, vibration\_count  count = (count if isinstance(count, Number) else 0) + 1  pass start.pressed(start\_pressed)  def touch\_button0\_pressed():  global count, seconds, hours, minutes, vibration\_count  touch\_button0.set\_hidden(True)  vibration\_count = (vibration\_count if isinstance(vibration\_count, Number) else 0) + 1  power.setVibrationEnable(False)  pass touch\_button0.pressed(touch\_button0\_pressed)   seconds = 0 minutes = 0 hours = 0 count = 0 vibration\_count = 1 touch\_button0.set\_hidden(True) while True:  if count % 2 == 1:  if hours >= 1 and minutes == 0 and seconds == 0:  minutes = 59  seconds = 60  hours = (hours if isinstance(hours, Number) else 0) + -1  elif hours >= 1 and minutes >= 1 and seconds == 0:  seconds = 60  minutes = (minutes if isinstance(minutes, Number) else 0) + -1  elif hours == 0 and minutes >= 1 and seconds == 0:  minutes = (minutes if isinstance(minutes, Number) else 0) + -1  seconds = 60  elif hours == 0 and minutes == 0 and seconds == 0:  count = (count if isinstance(count, Number) else 0) + 1  seconds = (seconds if isinstance(seconds, Number) else 0) + 1  if vibration\_count % 2 == 1:  power.setVibrationEnable(True)  vibration\_count = (vibration\_count if isinstance(vibration\_count, Number) else 0) + 1  touch\_button0.set\_hidden(False)  seconds = (seconds if isinstance(seconds, Number) else 0) + -1  wait(1)  h.set\_text(str(hours))  m.set\_text(str(minutes))  s.set\_text(str(seconds))  wait\_ms(2) |

#### 1.4.4 Batterimätning

Bästa metoden att mäta batteri är med användning av bränslemätare för batterier men författarna valde att inte inkludera en sådan komponent på grund av brist på utrymme. Modellering och design beskrivs inom [1.4.6 Banddesign och hållare](#_nc91hljtfyyb). Författarna valde istället att använda en mindre noggrann metod vilket endast använder batteriets spänning. Författaren utgick utifrån en [tabell från Lygte-info](https://lygte-info.dk/info/BatteryChargePercent%20UK.html) [[16]](#footnote-15)

<https://stackoverflow.com/questions/56266857/how-do-i-convert-battery-voltage-into-battery-percentage-on-a-4-15v-li-ion-batte>

#### 1.4.5 Google Calendar

På grund av anledningar som beskrivs i [3.1 Problem](#_30yhiaydjzjr) så valdes det under projektets gång att skrota stora delar av de önskade funktionerna gällande koppling till en sekundär platform - telefonen. Därför gjordes undersökningar i detta stadie om någon av de specifikationerna kunde lösas på annat sätt. Det framkom att på grund Core 2:s begränsade alternativ gällande trådlös kommunikation och begränsade tillgängliga bibliotek (återigen se [3.1 Problem](#_30yhiaydjzjr) och [3.3 Slutsatser](#_bcs6f0kc97ef) för djupare förklaring) fanns det enbart möjlighet att bibehålla funktionerna för koppling till google calendar samt möjligtvis väder.

Med detta bestämt valdes vidare att Kommunikation till och från klockan skulle ske över MQTT protokollet, då detta verkade mest anpassningsbart och lättillgängligt.

##### 1.4.5.1 Mosquitto MQTT

MQTT är som tidigare beskrivet ett protokoll som bygger på att klienter kan publicera meddelanden som avlyssnas och läses av andra klienter. För kommunikation mellan klienter över MQTT krävs därför en mellanhand, en server. Mosquitto är ett öppet program som används för att skapa privata MQTTservrar.



Vidare är det viktigt att för resten av stycket veta MQTT använder tre olika nivåer av QoS - Quality of Service - för att bestämma om den ska skicka publicerade meddelanden till prenumererade max en gång, minst en gång eller strikt en gång. Dessa nivåer benämns som **q 0**, **q 1** och **q 2**.



###### 1.4.5.1.1. Installation av Mosquitto

För installation av Mosquitto på Windows 10, som användes i projektet, krävs användning av 7-Zip som i sin tur först behöver installeras och laddas ner. Ladda ner och öppna senaste versionen av programvaran [här](https://www.7-zip.org/download.html)[[17]](#footnote-16). Följ sedan instruktionerna på skärmen. Vidare behövs nedladdning av [anpassning av mjukvaran för Windows 10](http://www.steves-internet-guide.com/wp-content/uploads/mos.7z)[[18]](#footnote-17) samt [grundversion av mjukvaran](http://www.eclipse.org/downloads/download.php?file=/mosquitto/binary/cygwin/mosquitto-1.4.14-install-cygwin.exe)[[19]](#footnote-18). I det fall att den första länken för dig till en blank sida med texten *Redirecting you to http://www.steves-internet-guide.com/wp-content/uploads/mos.7z* klipp ut länken och klistra själv in i valfri webbläsare.

Grundversionen körs, och sedan följs stegen som installeraren visar. Rekkomenderat är att inte välja en annan plats för installationen. Istället bör den lämnas på *C:\Program File(x86)\mosquitto.*

###### 1.4.5.1.2 Test av Mosquitto

##### 1.4.5.2 Node.js

###### 1.4.5.2.1 Node Red

###### 1.4.5.2.2 Google Cloud Platform

##### 1.4.5.3 Överföring till Core 2

#### 1.4.6 Banddesign och hållare

## 

## 2. Resultat

### 2.1 Interna funktioner

### 

### 2.2 Internetkopplade funktioner

### 

### 2.3 Bandet och hållare

## 

## 3. Diskussion

### 3.1 Problem

### 

### 3.2 Erfarenheter

### 

### 3.3 Slutsatser

## 

## 4. Källförteckning

## 

## Bilagor





### 

# Projektplan

Macintosh HD:Users:LindaEllinore:Downloads:Malmostad_PAGY_liggande.jpg

Projektplan Gymnasiearbete Smartwatch

*Ett försök att skapa en billigare smartwatch baserat på den ESP32baserade M5 Stack Core-plattformen*

Gymnasiearbete

TE18TE

Teknikprogrammet

Pauliskolan

2020

Författare:

Hugo Strömberg

Niklas Ku

Handledare:

Axel Månsson

## Innehållsförteckning

[**Innehållsförteckning**](#_eo58gbkww1ma) **17**

[**1 Bakgrund**](#_9072421wd4z6) **19**

[**2 Syfte**](#_fmuuy9fo6bxc) **20**

[2.1 Specifikation](#_bx6h3vhvrpre) 20

[2.2 Experiment](#_twv52a4s1x2w) 21

[2.3 Resultat](#_gr1t292a9v10) 21

[**3 Metod**](#_jgde1arg9qfe) **22**

[3.1 Hårdvara](#_36ubxvm7wsfb) 22

[3.2 Mjukvara](#_sfegijg6d1v) 22

[**4. Bilagor**](#_7laumlsxfvlo) **23**

## 1 Bakgrund

Smartwatch, eller smartklocka, är en intelligent lösning på hur du i ett litet format kan kommunicera information till en användare. Klockan innehåller en enklare dator eller mikrokontroller och monteras på användarens arm.

Smartklockor har funnits i olika former sen sent 70-tal men har först på senare år tillgängliggjorts för en bredare användarbas. Projektets mål är att bygga en enkel och billig klocka av enkla komponenter som gemene man kan tillhandahålla, då de smartklockor som finns på marknaden till stor del fortfarande är dyra vilket gör dem otillgängliga för många.

# 

## 2 Syfte

### 2.1 Specifikation

Projektets mål är att skapa en smartwatch med förmågan att:

* Visa tiden - kan välja mellan analog och digital klocka som visas
* Kan ta tiden eller nedräkning
* Visa datum, samt en kopplad google-kalender
* Visa batteriprocent, datum och digital tid längst upp
* Ta emot notiser från telefon via bluetooth och signalera till ägaren med vibration eller ljud
* Visa notiserna på en skärm
* Ta emot samtal och ringa genom kontaktlista eller med digital numeriskt tangentbord genom mobilen - mest användbart om hörlurar redan är kopplade till mobilen
* Visa väder
* Visa batteri både på klockan och en kopplad telefon
* Kontrollera musikspelare på telefonen
* Spela något enklare spel, förslagsvis tetris
* Visa puls - värdena sparas
* Visa de olika funktionerna på varsin “panel” eller sida som är åtkomlig genom genvägar
* “Stör ej”-funktion för smartklockan och justera för mobil
* Skärmsläckare - kunna välja tidsperiod - ska spara batteri
* Nattläge

## 

### 2.2 Experiment

Experiment ska utföras för att etablera validiteten hos var och en av specifikationerna. Försök ska göras att få var och en av funktionerna att fungera som tänkt och detta ska sedan prövas för att se att det har uppfyllts. I första hand ligger projektets prioritet på själva klockan och dess mest grundläggande funktioner såsom att ta emot notiser och visa kalendern. I andra hand ska experiment och försök göras angående musikspelare och spel.

En extra funktion som läggs till fysiskt med smartklocka är en pulsmätare. Pulsmätaren kommer genomgå flera experiment och försök för att få den att fungera och ge pålitliga värden.

### 2.3 Resultat

Resultat för majoriteten av experimenten sker enbart i form av ett enkelt svar på huruvida det tilltänkta målet uppnåtts.

## 3 Metod

### 3.1 Hårdvara

Huvudprojekt

* [M5Stack Core 2](https://www.elfa.se/sv/esp32-m5core2-development-module-m5stack-k010/p/30181494?queryFromSuggest=true) - 2x 730 kr (exkl. moms)
* Laddare - inkluderat i köpet av ovanstående
* [Li-Po batteri](https://www.elfa.se/sv/laddningsbart-batteripaket-li-po-7v-510mah-renata-icp303450pa-02/p/30158706?q=*&pos=1&origPos=2&origPageSize=10&track=true) - 2x 338 kr (exkl. moms)
* [Pulssensor](https://www.elfa.se/sv/pulssensor-sparkfun-electronics-sen-11574/p/30152838?q=puls+sesnor&pos=6&origPos=6&origPageSize=10&track=true) - 2x 462 kr (exkl. moms)
* [Sladdar](https://www.kjell.com/se/produkter/el-verktyg/elektronik/elektroniklabb/luxorparts-delbar-kopplingskabel-40-pol-hona-hona-p87906) - 90 kr
* [Batteri för Core basic 155 mAh](https://www.elfa.se/sv/laddningsbart-batteripaket-li-po-7v-155mah-renata-icp402025pc-01/p/30158709?q=*&pos=5&origPos=13&origPageSize=10&track=true) - dessutom kommer ett batteri till den inlånade basic core införskaffas då den var svullen- 124 kr (exkl. moms)

Kostnad:

Elfa: Beställning 1 (batterier): 338+124+105(frakt)+142(moms) = 709 kr ≈ 710 kr

Beställning 2 (Core2+HR sensor): 730+462+0(frakt)+298(moms) = 1490 kr

Kjell: 90 kr

Övrigt: 10 kr

Totalt: 2 300 kr

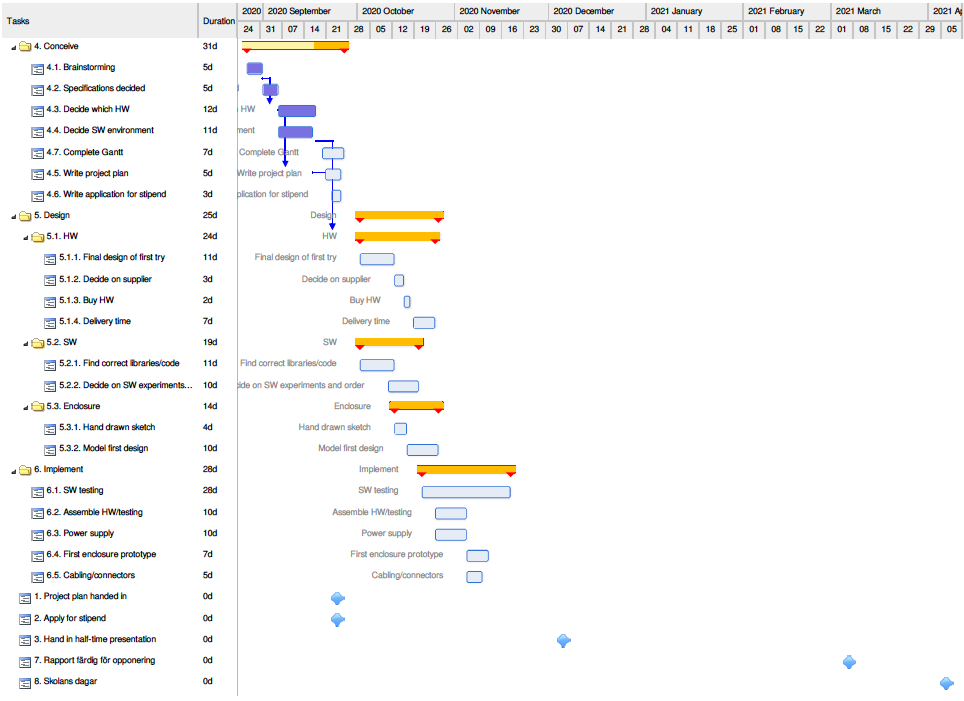
### 3.2 Mjukvara

Ingen mjukvara kommer att inköpas för projektet. Den mjukvara som används är antingen gratis eller open-source och kostar därför inget.

Till största del kommer programmeringen ske i Visual Studio Code på språket C ++, inom Arduino IDE miljön. Exempelkod och

## 4. Bilagor

Ganttschema



1. "ESP32 Wi-Fi & Bluetooth MCU I Espressif Systems." <https://www.espressif.com/en/products/socs/esp32>. Accessed 26 Feb. 2021. [↑](#footnote-ref-0)
2. "M5Stack Core2 ESP32 IoT Development Kit | m5stack-store." <https://m5stack.com/products/m5stack-core2-esp32-iot-development-kit>. Öppnades 26 feb.. 2021. [↑](#footnote-ref-1)
3. "UIFlow | m5stack-store." <https://m5stack.com/pages/uiflow>. Öppnades 26 Feb. 2021. [↑](#footnote-ref-2)
4. "Update firmware · UIFlow Use." <https://m5stack.github.io/UIFlow_doc/en/en/base/Update.html>. Öppnades 26 Feb. 2021. [↑](#footnote-ref-3)
5. "Eclipse Mosquitto." <https://mosquitto.org/>. Öppnades 26 Feb. 2021. [↑](#footnote-ref-4)
6. "7-Zip." <https://www.7-zip.org/>. Öppnades 26 feb.. 2021. [↑](#footnote-ref-5)
7. "Node.js." <https://nodejs.org/>. Öppnades 26 feb.. 2021. [↑](#footnote-ref-6)
8. "Node-RED." <https://nodered.org/>. Öppnades 26 Feb. 2021. [↑](#footnote-ref-7)
9. "Google Cloud." <https://cloud.google.com/>. Öppnades 26 Feb. 2021. [↑](#footnote-ref-8)
10. "Realtidsklocka – Wikipedia." <https://sv.wikipedia.org/wiki/Realtidsklocka>. Öppnades 26 Feb. 2021. [↑](#footnote-ref-9)
11. "K010 | M5Stack ESP32 M5Core2 utvecklingsmodul | Elfa Distrelec ...." <https://www.elfa.se/sv/esp32-m5core2-utvecklingsmodul-m5stack-k010/p/30181494>. Öppnades 26 feb.. 2021. [↑](#footnote-ref-10)
12. "SEN-11574 | SparkFun Electronics Pulssensor | Elfa Distrelec Sverige." <https://www.elfa.se/sv/pulssensor-sparkfun-electronics-sen-11574/p/30152838>. Öppnades 26 feb.. 2021. [↑](#footnote-ref-11)
13. "ICP303450PA-02 | Renata Laddningsbart batteripaket, Li-Po ... - Elfa." <https://www.elfa.se/sv/laddningsbart-batteripaket-li-po-7v-510mah-renata-icp303450pa-02/p/30158706>. Öppnades 26 feb.. 2021. [↑](#footnote-ref-12)
14. "800cc Onyx Filament Spool | Markforged." <https://markforged.com/product/onyx-filament/>. Öppnades 26 feb.. 2021. [↑](#footnote-ref-13)
15. "WiFiConfig - Arduino." <https://www.arduino.cc/en/Reference/WiFiConfig>. Öppnades 26 feb.. 2021. [↑](#footnote-ref-14)
16. "Battery charge percent - Lygte-info.dk." <https://lygte-info.dk/info/BatteryChargePercent%20UK.html>. Öppnades 26 feb.. 2021. [↑](#footnote-ref-15)
17. "Download 7-Zip." 21 feb.. 2019, <https://www.7-zip.org/download.html>. Öppnades 26 feb.. 2021. [↑](#footnote-ref-16)
18. "Steve's Internet Guide." <http://www.steves-internet-guide.com/wp-content/uploads/mos.7z>. Öppnades 26 feb.. 2021. [↑](#footnote-ref-17)
19. "Download | Eclipse Mosquitto." <https://mosquitto.org/download/>. Öppnades 26 feb.. 2021. [↑](#footnote-ref-18)