## Dokumentation – Reaktionbrädet Upptech 2024

**Komponenter:**

Raspberry PI 4 1GB

1x RGB LED Matrix Panel Drive Board For Raspberry Pi (Hat)

* **SKU:** MPC1073
* https://www.electrodragon.com/product/rgb-matrix-panel-drive-board-raspberry-pi/

2x 128x64-LCD-skärmar + Strömkablar och kablar till Hatten

Raspberry PI Pico WH

Strömkälla till LCD-skärmar 5V 40A

3x LED röd

3x 220 Ohm motstånd

**Funktion Pi4 och Pi mm:**

För att starta spelet så kopplar du enbart in Tre kablar till förslagsvis ett förgreningsuttag. Då skall brädet lysa upp och första spelet startar. Notera att under första start efter avbrott i strömförsörjningen så visar den gamla poängräknaren 1 poäng. Detta är inget att oroa sig för.

Pi4:an tar cirka 30 sekunder för att starta vid inkoppling och när den väl har startat så körs ett script som helt enkelt startar programmet view.py. Detta program beskrivs nedan. När Pi4:an blir försörjd med el så startar Picon och denna börjar då ta emot signalerna från brädet. Detta sker alltså innan LCD-skärmarna har startat.

**Pi4:an:**

Pi4:ans syfte är att strömförsörja Picon, tar emot signalerna från denna och styr vad som ska printas på LCD-skärmarna. Man kan säga att Pi4:an är presentationslagret.

Signalen som kontinuerligt skickas från Picon en gång per sekund eller när giltig grön/röd knapp trycks ned är en sträng med formatet **<poäng|tid> eller <-poäng|tid>**, notera minustecknet i början av den andra strängen. Detta används för att signalera att spelet är slut och att det är dags att kolla om highscoren behöver uppdateras. Om spelets nuvarande poäng är större än något på tavlan så ersätts detta.

**Pico:**

Picon är mer eller mindre Logiklagret i systemet och tar emot signalerna från brädet, interpreterar dessa så att rätt poäng och tid skickas över till Pi4:an, när Picons inre spelklocka når noll så lägger den till ett minustecken i början av strängen som den skickar över till Pi4:an, ignorerar alla eventuella inkommande signaler förutom från den röda resetknappen och ”E-signalen”.

Picon skickar över en sträng **<poäng|tid>** eller **<-poäng|tid>** kontinuerligt varje sekund men också då den detekterar en valid signal från den gröna eller röda knappen.

**Kablar mellan Picon och reaktionsbrädets logikkrets:**

Den gula kabeln **utan** eltejp är kopplad till reaktionsbrädets R-signal. Alltså den röda resetknappen. Denna signal är hög när den inte är aktiv men låg när den är nedtryckt.

Den gula kabeln **med** eltejp är kopplad till brädets e-signal och har ingen funktion i mjukvaran men den signalerar för brädets logikkretsar att spelet är över genom att gå hög då spelet ej är i gång. E-signalen är alltså Hög när spelet är i gång. Denna kabel är skarvad med en svart kabel som går in i brädet.

Den gröna kabeln är kopplad till brädets S-signal som signalerar ifall rätt grön knapp har blivit tryckt. Denna signal är hög när den inte är aktiv men låg när den är nedtryckt.

Den svarta kabeln är kopplad mellan brädet och Picons jord. Syftet är att koppla samman jordarna. Utan denna blir spelet mycket lätt instabil.

**Mjukvarukomponenter:**

**Pico:**

Main.py

**Pins:**

* led\_internal: Slås av och på vid varje loop.
  + Indikerar att Picon är på
* grer\_button: GPIO15, Insignal, Pull-Up resistor
  + Poänsignal
* red\_button: GPIO16, Insignal, Pull-Up resistor
  + Resetsignal
* is\_finished: GPIO17/18, Insingal/Utsignal
  + Ingen funktion för tillfället

**Globala Variabler:**

score: Antalet poäng, startar på noll och sätts till noll vid reset

time\_remaining: Tid kvar, startar på 60 och sätts till 60 vid reset

tim: Vilken timer på Picon som används

time\_last

time\_current

have\_sent\_highscore: Falsk vid start, sätts till Sann när spelet är över och sätts till Falsk vid reset

**Callbackfunktion och klocka:**

Det centrala I denna fil är klockan som kallar callbackfunktionen timer\_callback en gång per sekund. I denna funktion så kollas först att tiden inte är lika med noll. Om tiden inte är lika med noll så subtraheras time\_remaining med ett. Därefter kollar funktionen om time\_remaining är lika med noll. Om tiden är lika med noll så är spelet slut och då skall Picon skicka ut till brädet att E är hög, detta används ej för tillfället. Därefter kollar funktionen om have\_sent\_highscore är falskt. Om have\_sent\_highscore är Falskt så läggs ett minustecken till i början av strängen som skickas över till Pi4:an, detta för att indikera spelets slut. Strängen skickas sedan över till Pi4:an och have\_sent\_highscore sätts till Sann. Vid varje kall på denna funktion så kallas funktionen send\_data med nuvarande poäng och kvarvarande tid.

Timern är -1 timern

**timer\_init()**

Initialiserar timern som har en frekvens på 1000 och en period på 1000 vilket ger klocka som tickar varje sekund.

**send\_data()**

Skickar över data till Pi4:an som en sträng

**reset\_game()**

Återställer spelet och dessa variabler till dess standardvärden.

**grer\_button\_interrupt(grer\_button)**

Vid varje fallande flank från den gröna knappen så aktiveras denna interrupt, den kollar först så att den senaste fallande flanken hände för över 50ms. Detta för att filtrera ut kontaktstuds. Sedan kollar den ifall spelet är i gång och om spelet är igång så ökas poängen och send\_data kallas

**red\_button\_interrupt()**

Vid varje fallande flank från den röda knappen så aktiveras denna interrupt, den kollar först så att den senaste fallande flanken hände för över 500ms. Detta för att filtrera ut kontaktstuds. Efter detta kallas reset\_game funktionen

**Pi4:**

**notes.txt**

Ett par anteckningar som är bra ha på Pi4:an

**view.py**

**Globala Variabler och Konfiguration:**

Noto\_Serif och Noto\_Sans är typsnitt.

Under if-satserna under dessa så kommer självaste konfigurationen till LED-displayerna.

**Funktioner:**

**image\_draw:** Denna funktion får in strängen från Picon och omvandlar den till en bild som kan printas på skärmarna.

Den kollar först ifall det finns ett minustecken på index 0. Om det finns ett minustecken på index 0 så kollar denna ifall det nuvarande scoret är högre än något i listan. Om det är högre, så tas det sista värdet ut och poängen läggs in på rätt plats i listan.

Därefter så returnerar den bilden som skall printas på skärmarna. Sedan kollar funktionen om bilden är bredare än 256 pixlar. Om detta är sant så förflyttas bilder i x-led. Bilden printas sedan till LED-skärmarna

**print\_text:** Kallar först på funktion image\_draw för att inhämta bilden som printas skall.

**get\_text:** Inhämtar strängen från porten ttyACM0 och formaterar denna till ett format som är läsbart för oss.

RPI\_PICO\_W-2024022-V1.22.2.UF2 – Picons “configfil”