Turrisem řízené hodiny

Ondřej Caletka

29. listopadu 2014

PROJECT:TURRIS

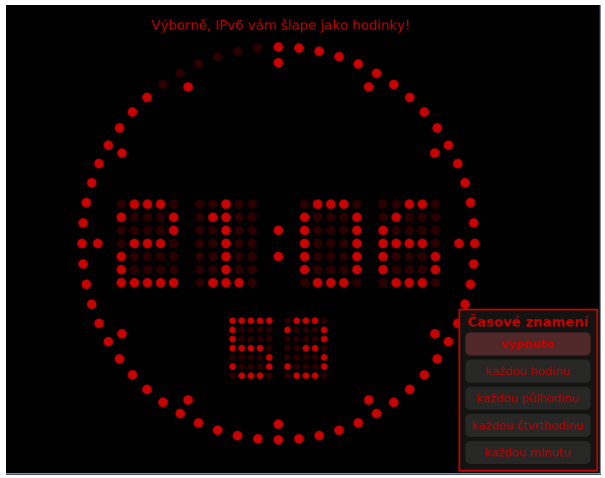
Kdo je Ondřej Caletka

- absolvent ČVUT FEL
- zaměstnanec sdružení CESNET
- uživatel routeru Turris
- zájmy:
  - Linux a open source
  - počítačové sítě a IPv6
  - elektrotechnika a elektronika
  - hromadná doprava („*šotouš*“)
  - **přesný čas** (~50 ms)

PROJECT:TURRIS

Reference z oblasti přesného času

- český překlad serveru Time.is
- autonomní NTP server s přijímačem DCF77
- studiové HTML5 hodiny čas.Neběží.cz
- Turrisem řízené hodiny



PROJECT:TURRIS

Proč nepoužít rádiem řízené hodiny

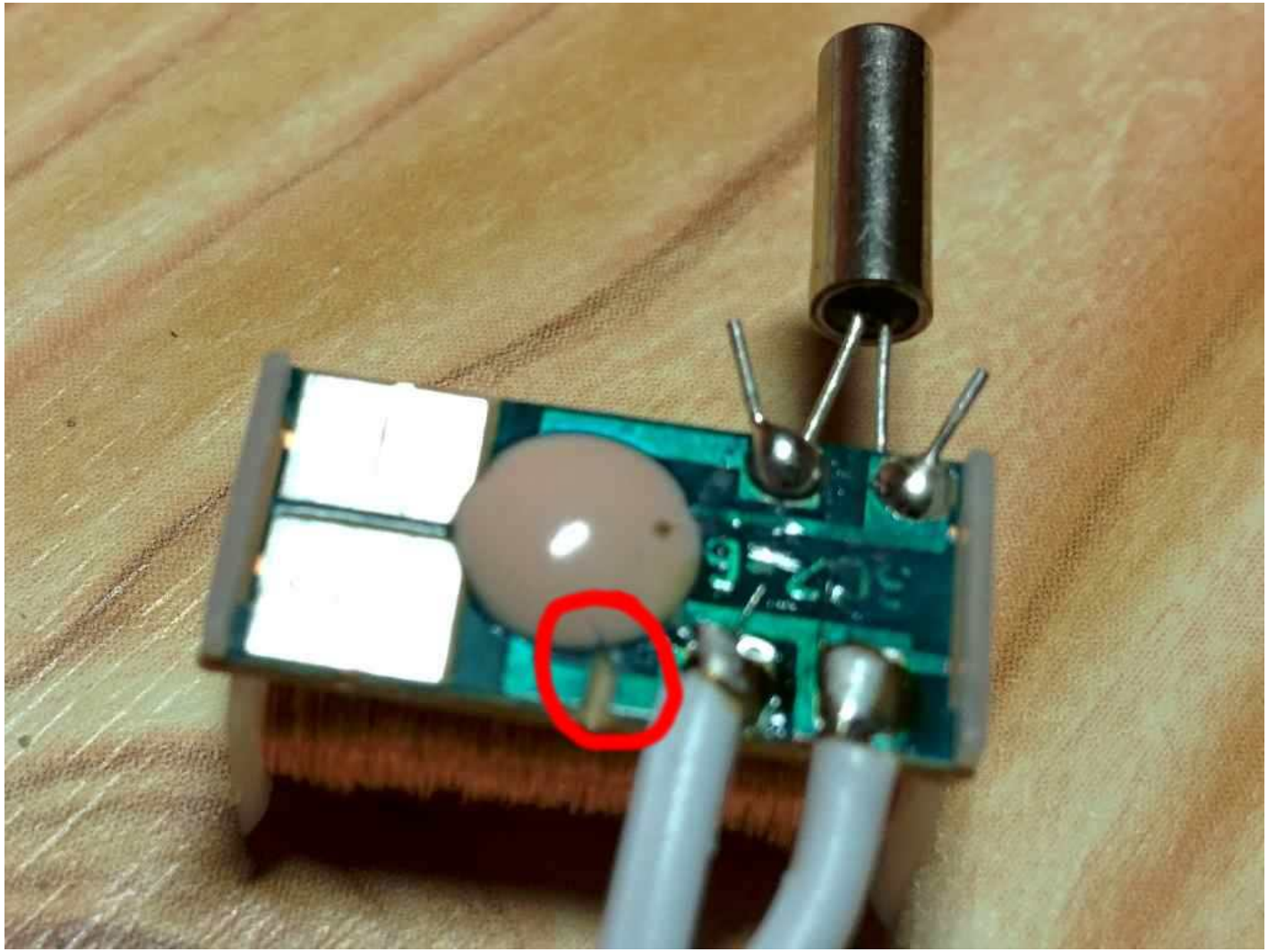
- malý výběr, velké náklady
- špatně fungují uvnitř budov
- synchronizace pouze 1× denně

Výhody Turrisem řízených hodin

- neustále synchronní díky NTP
- nepotřebují baterie
- je možné je zastavit např. na noc

PROJECT:TURRIS

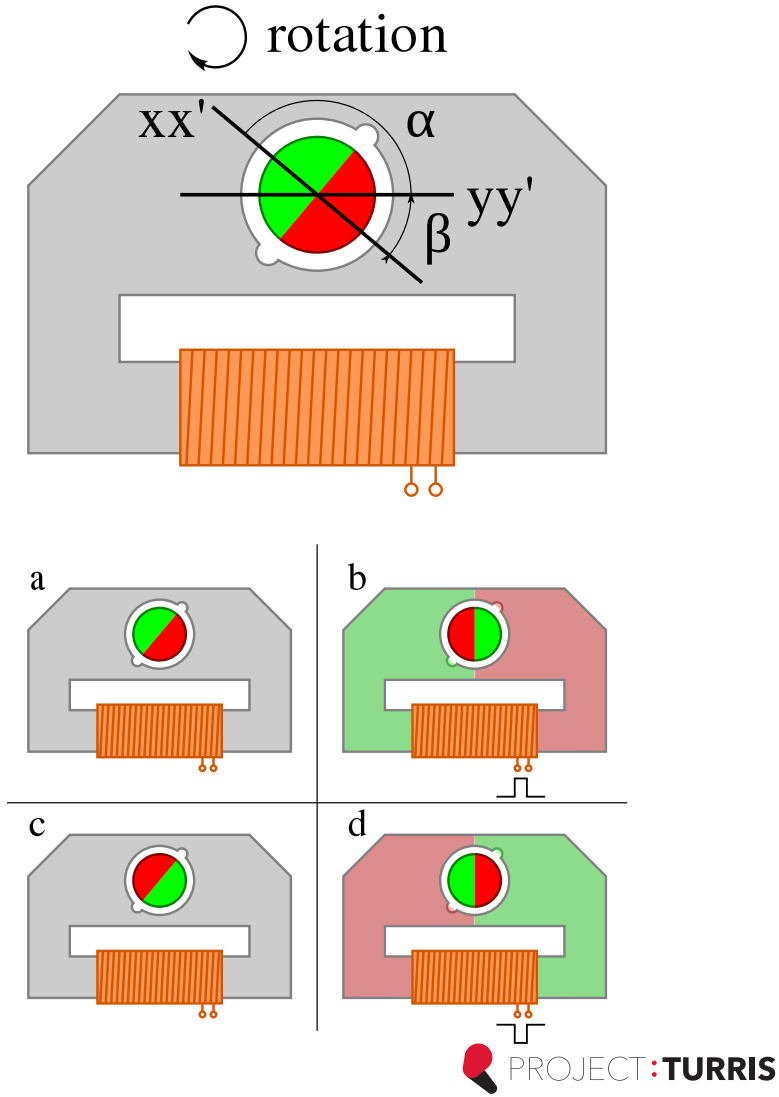
Úprava hodin



PROJECT:TURRIS

Lavetův krokový motor

- Marius Lavet, 1936
- pevná klidová poloha
- neměnný směr otáčení
- napájení polarizovanými impulzy
- široce využíván v elektrických hodinách od quartzové revoluce (~1970)



PROJECT:TURRIS

Hardwarový interface

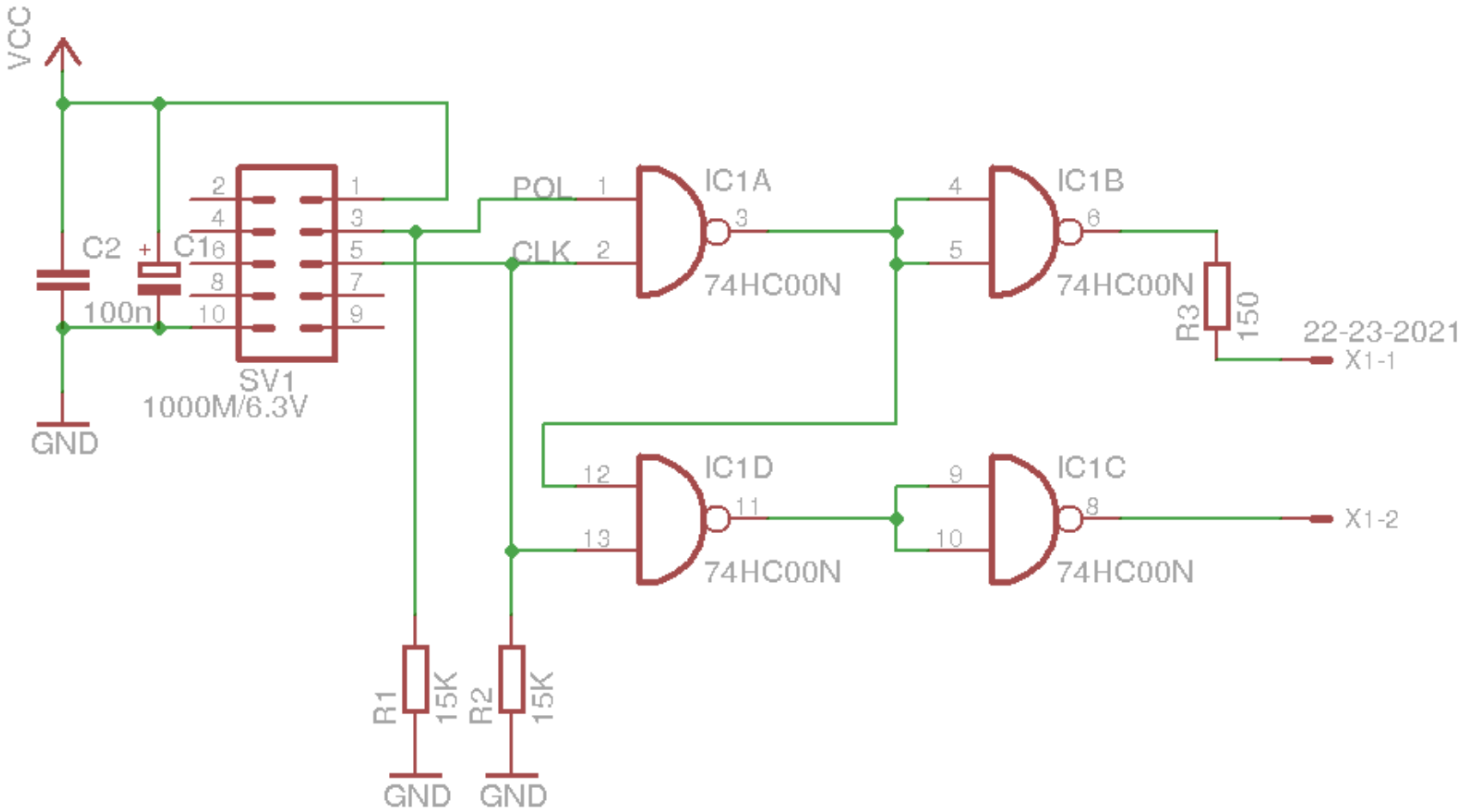
- jednoduchý budič s obvodem 74HC00
- připojuje se ke GPIO konektoru Turrise
- dva ovládací GPIO piny

  POL určuje polaritu výstupního napětí

  CLK povoluje napájení lavetova motoru

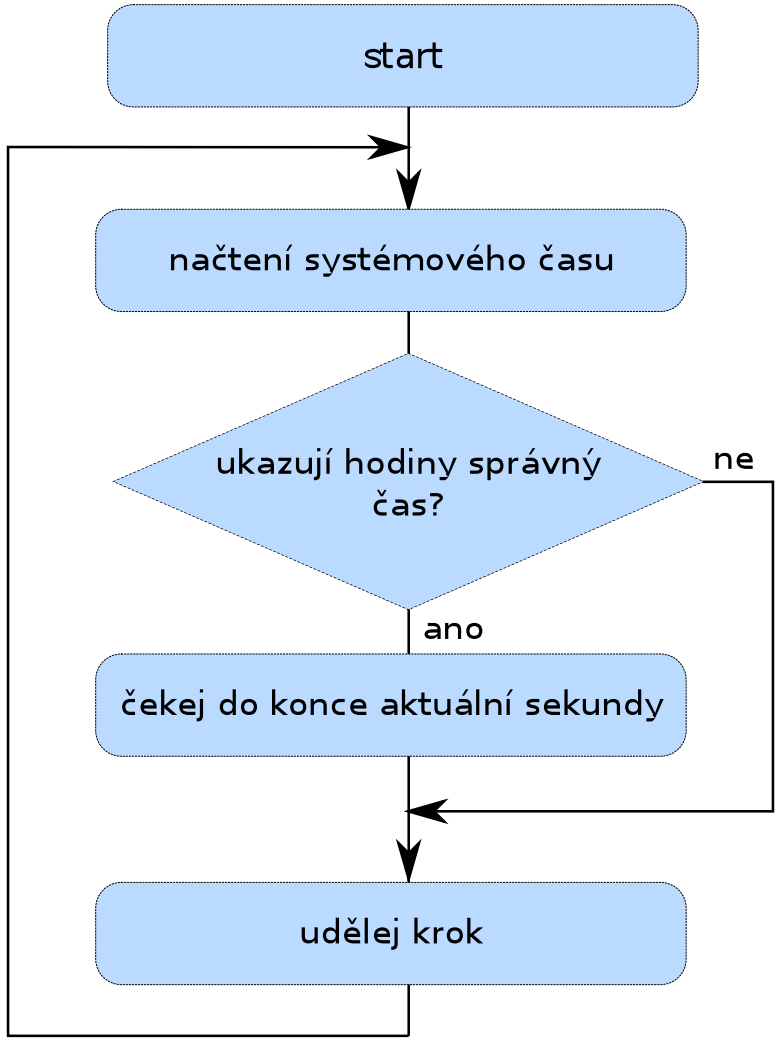
PROJECT:TURRIS

Zapojení interface



Software

- napsaný v Pythonu 2
- ovládá GPIO výstupy pomocí sysfs interface
- udržuje v paměti aktuální stav ručiček
- volí správně polaritu impulzů
- používá systémový čas a funkci `sleep`
- snaží se chránit GPIO před poškozením



PROJECT:TURRIS

Perzistence stavu ručiček

- poprvé musí uživatel odečíst aktuální stav ručiček
- ten se drží v paměti po dobu běhu hodin
- při korektním ukončení je uložen do souboru
- při nekorektním ukončení je nutný zásah uživatele

Kam ukládat průběžný stav?

- ✗ RAM disk – neperzistentní
- ✗ Flash paměť – omezený počet zápisů
- ✓ RTC SRAM – ideální, ale chybí podpora v driveru ☹

PROJECT:TURRIS

Jdou hodiny napřed nebo pozdě?

- obojí; záleží jen na úhlu pohledu
- rychlý posun vpřed nemusí být nejrychlejší způsob, jak odchylku hodin srovnat
- před každým krokem se vyhodnocuje, zda je výhodnější pokračovat vpřed, nebo čekat
- čekání je přerušováno „komfortními kroky“ každých 10 sekund pro uklidnění uživatele

$$t_{fastforward} = \sum_{k=1}^{\infty} n_{diff} \cdot t_{step}^{k} = n_{diff} \cdot \frac{t_{step}}{1 - t_{step}} \tag{1}$$

$$t_{wait} = \sum_{k=0}^{\infty} n'_{diff} \cdot \left(\frac{1}{t_{comfort}} \right)^{k} = n'_{diff} \cdot \frac{t_{comfort}}{t_{comfort} - 1} \tag{2}$$

PROJECT:TURRIS

Praktické ukázky

Běžný provoz

```
./turrisclock.py --state 13:40:23
```

Krokování po delších intervalech

```
./turrisclock.py --step 10
```

Vyčkávání do času s komfortními kroky

```
./goto.py 13:50:00
./turrisclock.py
```

PROJECT:TURRIS

Další vývoj

- použít RTC k perzistentnímu ukládání stavu ručiček
- vylepšit ovládání GPIO
- daemonizace kódu
- navázání na události v Turrisu
- *přepis do C*

Kompletní dokumentace HW i SW je k dipozici:
`http://oskar456.github.io/turrisclock/`

PROJECT:TURRIS

Závěr

Děkuji za pozornost

Ondřej Caletka
mailto:Ondrej@Caletka.cz
http://Ondřej.Caletka.cz

PROJECT:TURRIS