Přístrojová a řídicí technika

Projekt 3

Grafické rozhraní pro ovládání měřící karty LabJack U3-HV

Vedoucí práce: doc. Bukovský

Vypracoval: Martin Fayad

Obsah

Prostředky pro tvorbu grafického rozhraní

Nejvhodnějším prostředkem pro tvorbu grafického rozhraní pro měřící kartu LabJack jsem zvolil Python 2.7. Jednak pro python existuje knihovna LabJackPython, se kterou lze poměrně jednoduše ovládat měřící kartu a jednak jsem chtěl vyzkoušet nepříliš známou, nicméně velmi zajímavou knihovnu na tvorbu grafického rozhraní Kivy. S Kivy lze vytvořit aplikace pro většinu operačních systému a to jak na stolní počítače tak i pro mobilní zařízení. Nabízela se tudíž zajímavá myšlenka vytvoření aplikace pro ovládání Labjacka z počítače, kterou by bylo možné spustit i na mobilním zařízení.

Kivy má velmi jednoduchou a však mocnou syntaxi, díky čemuž lze velmi rychle vytvořit aplikace pro své programy. Ačkoliv Kivy není primárně určeno pro vědecké ani inženýrské účely, lze jej právě pro tyto oblasti využít. Kivy má moderním vzhled a funkce (podpora dotyku, interaktivní) a její licence povoluje komerční využití.

V tomto Projektu bych chtěl demonstrovat potenciál Kivy při tvorbě grafického rozhraní pro měřící kartu LabJack

Instalace

Instalace knihoven do Pythonu často bývá problematická, proto zde uvedu krok po kroku instalace všech knihoven a modulů, na kterých bude záviset Aplikace pro LabJack

Windows

Instalace Pythonu 2.7.11

Ze stránek Python.org stáhnout instalační soubor pro Python 2.7.11 - 32/64 bit a nainstalovat jej.

Pip

Stáhnout a nainstalovat modul pro python „Pip“, se kterým lze instalovat knihovny do Pythonu.

- Stáhnout soubor get-pip.py, a příkazem v CMD **‚py -2 get-pip.py‘** jej nainstalovat

Instalace Kivy

Instalace závislých knihoven.

Příkaz v CMD:

“py -2 -m pip install docutils pygments pypiwin32 kivy.deps.sdl2 kivy.deps.glew”

Instalace modulu gstreamer:

Stáhnout gstreamer pro správnou verzi Pythonu

-kivy.deps.gstreamer-0.1.7-cp27-cp27m-win_amd64.whl

nebo

-kivy.deps.gstreamer-0.1.7-cp27-cp27m-win_amd32.whl

Příkaz v CMD:

„py -2 -m pip install kivy.deps.gstreamer-0.1.7-cp34-cp34m-win_amd32.whl“

„py -2 -m pip install kivy“

Příkaz v CMD:

„cd C:\Python27\Scripts“

A dále

„garden install matplotlib“

Instalace ostatních knihoven

Příkaz v CMD:

„py -2 -m pip install numpy, matplotlib, pickle“

Instalace LabJackPython

Ze stránek **https://labjack.com/support/software/examples/ud/labjackpython**, stáhnout složku a příkazem v CMD **„py -2 setup.py install“** se nainstaluje knihovna pro ovládání LabJacka

Dále je potřeba stáhnout a nainstalovat LabJack ovladač na Windows, který lze nalézt na stránce **https://labjack.com/support/software/installers/ud**.

Linux

Postup je vcelku stejný. Na Linuxu je již Python společně s Pip modulem nainstalovaný.

Ovládání LabJacka přes LabJackPython knihovnu

Pro ovládání LabJacka pomocí LabJackPython knihovny se využívá LabJack Modbus rozhraní, s kterým se zapisuje do registrů. Tento způsob je celkem komplikovaný a nepřehledný a proto jsem napsal krátkou knihovnu se kterou lze jednodušeji a intuitivněji používat LabJacka.

Kód této knihovny bude uveden v příloze.

Příklady pro ovládání vstupů a výstupů LabJacka pomocí mé knihovny:

Vytvořím objekt:

import LJcontrol as LJ

cl = LJ.controll()

```
# Analog output na portu 0, výstup 3.12V
cl.AO(cl.DAC['0'],3.12)

# Analog INPUT na portu 3, čtení vstupu uloženo do proměnné HODNOTA
HODNOTA = cl.AI(cl.AIN['3'])

#Digital Output, výstup logická 1, 3.3V
cl.DO(cl.FIO['7'],1)
#Digital Input , získaní informace o tom, zda do portu 3 jde signál
HODNOTA = cl.DI(cl.FIO['3'])

#Digital Input Read Only , tato funkce pouze čte logickou hodnotu na portu.
HODNOTA = cl.DI_RO(cl.FIO['5'])
```

Data z měření

Program jsem navrhl tak, aby se dalo měřit na vzdáleném zařízení a výsledky měření se pošlou bezdrátově přes síť na zařízení, kde je řídicí aplikace. Díky tomuto návrhu lze například řídit LabJack zapojený v Raspberry Pi, který bude přijímat a odesílat data na počítač či mobilní zařízení.

Princip přenosu dat

Přes síť se posílají 2 python objekty – slovníky.

První se posílá z aplikace (Klient) do zařízení (Server), které ovládá LabJacka – např. Raspberry pi, nebo laptop. Tento objekt obsahuje příkazy, vytvořené uživatelem a server je po přijetí interpretuje a vykoná.

Druhý objekt se posíla ze zařízení (Server), které ovládá LabJacka do zařízení, kde je Aplikace (Klient). Tento objekt obsahuje hodnoty ze vstupů LabJacka.

Příklad vytvoření serveru

Pomocí této funkce establish() se vytvoří server, který posílá i přijímá data. Pokud se klient odpojí, může se později opět připojit. Server běží v nezávislým vláknu, proto může zbytek programu plynule fungovat.

```
def establish(): ## Establish server

        import socket
         #### nalezeni vlastni IP - NA TUTO IP SE PAK PRIPOJIME Z APLIKACE #####
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        s.connect(("gmail.com", 80))
        MY_LOCAL_IP = (s.getsockname()[0])
        s.close()

        ###################################
        s = socket.socket()
        port=10011 #### Pouzijeme tento PORT
        s.bind((str(MY_LOCAL_IP), port))
        s.listen(5)

        while True:
            c, addr = s.accept()
            print('Got Connection from:', addr)
            while True:
                global controller
                try: ## PRIJMU DATA,ROZBALIM POMOCI PROTOCOLu 2
                    received_data = c.recv(1024*350)
                    controller = pickle.loads(received_data)
                except IndexError:
                    print('index error')
                except:
                        break

                try     ## ODESLU DATA, ZABALIM POMOCI PROTOCOLU 2
                        serialized_data=pickle.dumps(data_to_send,protocol=2)
                        c.send(serialized_data)
                except:
                        pass

       #### Necham server bezet v samostatnem vlakne

threading.Thread(target=establish).start()
```

Aplikace

V aplikaci napsané v Kivy jazyce jsem se snažil využít předností této knihovny a vytvořil jsem interaktivní ovládání výstupů, které by na počítači (Windows) s dotykovým displejem mělo být ovladatelné dotykově.

Podařilo se mi do Kivy naimportovat Matplotlib knihovnu, kterou jsem přizpůsobil, ačkoliv k tomu není vhodná, na realtime zobrazovaní dat s celkem vysokou rychlostí zobrazování. Je tedy možné sledovat v jednom okamžiku dění na všech vstupech.

Návod na používání aplikace

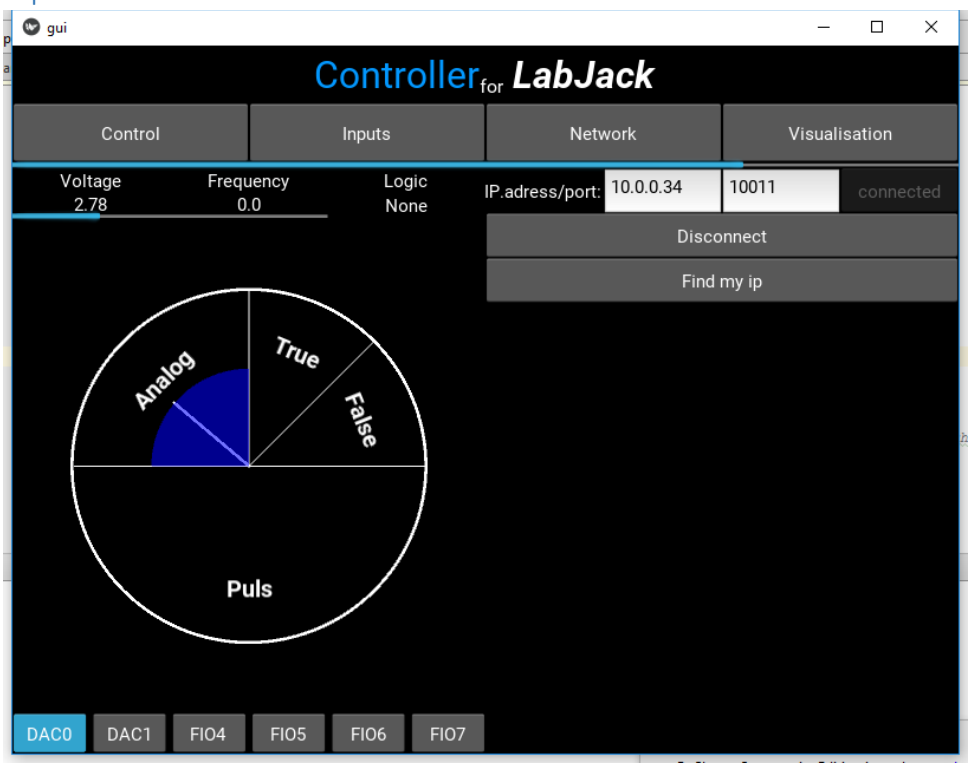
1) Zařízení, který slouží jako Server musí mít nainstalované knihovny a driver na LabJacka.
2) Ve složce se skriptem, který zakládá server musí být též soubor LJcontrol.py, ve kterém je má knihovna na ovládání LabJacka.
3) Po vytvoření serveru se lze ze zařízení, na kterém je aplikace, připojit na server. Je potřeba znát IP adresu serveru a port.
4) V případě, že se LabJack připojí na zařízení, ze kterého se i bude řídit (Aplikace), musíme do stejné složky jako je aplikace umístit skript se serverem (control.py) i moji knihovnu LJcontrol.py a na úplný začátek aplikace(main.py) připsat

```
def python_sc():
    import os
    os.system('py -2 control.py')
threading.Thread(target=python_sc).start()
```

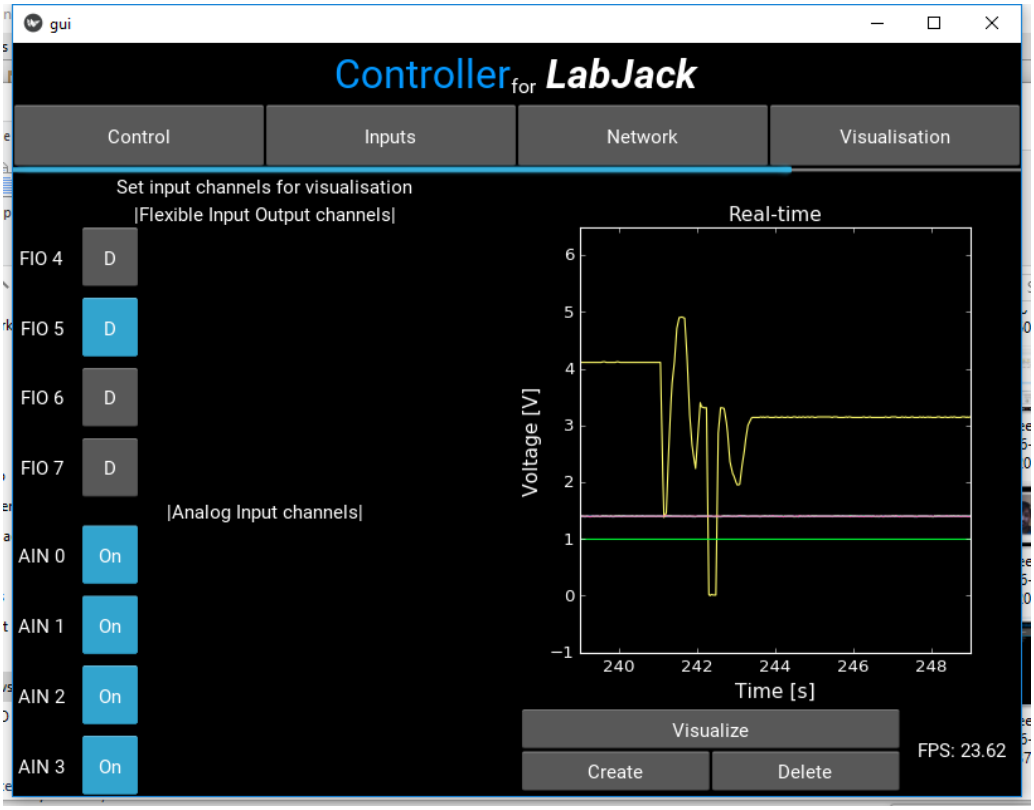
nebo tento skript control.py spustit ručně.

5) Pokud jsou všechny body splněny, stačí mít LabJacka zapojeného v zařízení a lze jej začít ovládat prostřednictvím Aplikace .

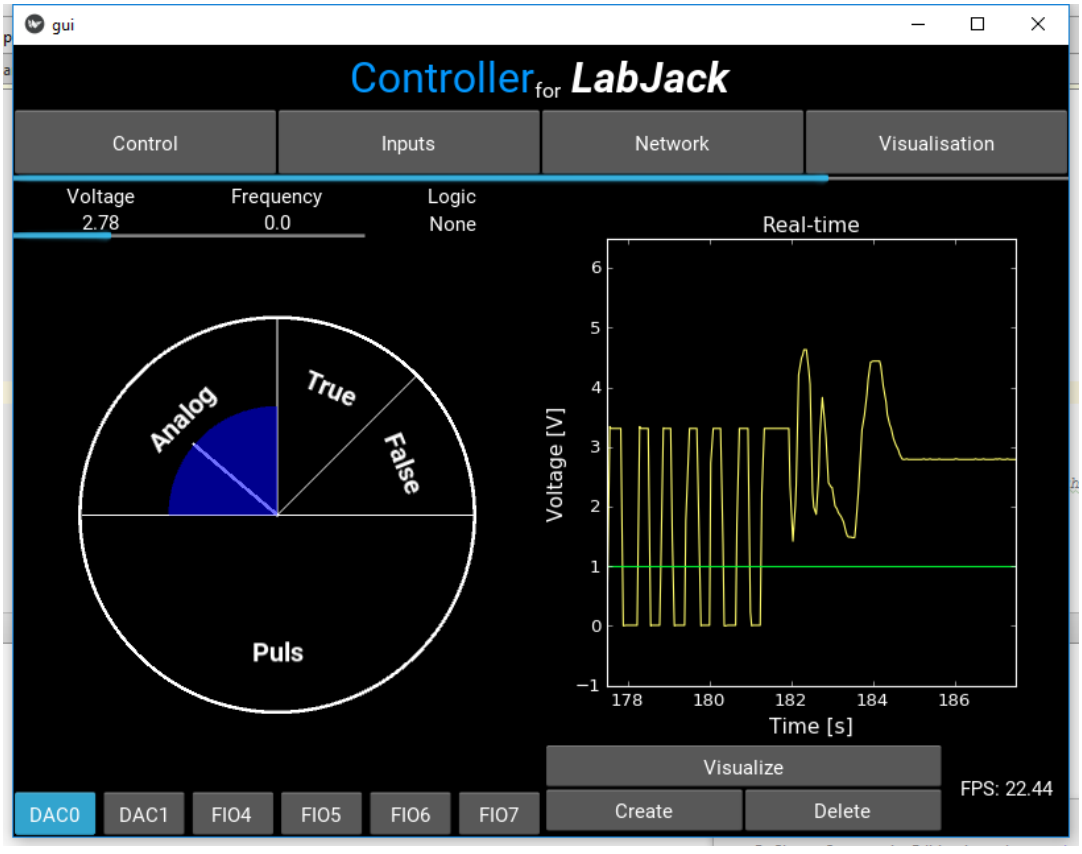
Ukázky z Aplikace



Obrázek 1: Vlevo - Interaktivní ovládání, vpravo – panel na připojení k serveru



Obrázek 2: Vlevo - nastavení vstupů, Vpravo - Realtime měření ze vstupů



Obrázek 3: Pomoci 'Puls' lze vytvářet pulzní signál

Kód

Aplikace má celkem 4 soubory.

1) control.py – vytvoří server a tento soubor se nachází vždy na zařízení, kde je zapojen LabJack
2) LJcontrol.py – je má knihovna, která usnadňuje ovládání LabJacka. Musí být ve stejné složce jako je control.py
3) gui.kv – je textový soubor, který vytváří strukturu grafického rozhraní a je napojen na hlavní soubor aplikace.
4) main.py je soubor, který spouští aplikaci. Je-li gui.kv soubor vytvářející strukturu aplikace, pak main.py je soubor, který vytváří celou logiku.

Celkem mají tyto souboru přes 800 řádku a proto zde neuvedu kód, ale odkaz, kde se soubory nacházejí.

Zde lze stáhnout soubory Aplikace.

https://github.com/marfay/LabJack-U3-HV-Control

Aplikace v mobilním zařízení

Ačkoliv to nebylo cílem mého projektu, při pokusu spustit aplikaci na mobilním zařízení s Androidem jsem se setkal s částečným neúspěchem. Aplikaci se mi nepodařilo spustit, neboť naimportování aplikace, která má mnoho modulů, jako je například graf knihovny Matplotlib, je velmi komplikované. Nicméně se mi podařilo vytvořit velmi jednoduchou aplikaci, která pomocí tlačítek ovládá výstupy LabJacka. Velmi zajímavé je, jak jednoduchým způsobem lze vytvořit aplikaci spustitelnou na Androidu. Krátce zde uvedu postup, jak na to.

1) Na Androidu nainstalovat aplikaci Kivy Launcher
2) Na SD kartě nebo v paměti mobilního zařízení **nalézt** složku kivy.
3) Do složky kivy umístit složku s vaší aplikací.
4) Ve složce vaší aplikace musí být 3 soubory – main.py , gui.kv a android.txt
5) android.txt musí obsahovat 3 řádky:
   title=LabJack Control
   author=Martin Fayad
   orientation=portrait (nebo landscape)
6) Po spuštění aplikace Kivy Launcher by měla být vidět vaše aplikace

Závěr

Můj cíl bylo vytvořit grafické rozhraní, které umožňuje ovládat zařízení, v tomto případě LabJacka, pomocí pythonovské knihovny Kivy, která není příliš rozšířená. Důležité však pro mě bylo uplatnit Kivy na inženýrské aplikace. Mnoho knihoven na grafické rozhraní je buďto zastaralých nebo jsou pod omezující licencí. Kivy ač není primárně určena na inženýrské aplikace se v mém Projektu osvědčila a dle mého názoru je perfektní jako inženýrský nástroj při uskutečňování řešení.

Aplikace, tak jak jsem ji vytvořil je kompletní, avšak není dokonalá. Je zde mnoho, co by se dalo zdokonalit nebo dokonce přidat. Díky jednoduchosti Kivy může kdokoliv tuto aplikaci posunout dál.