CASE STUDY

Practical data connection between MATLAB and microcontrollers using virtual serial port and MicroPython Pyboard: A survey

Ashkan Safari | Mehran Sabahi

Faculty of Electrical and Computer Engineering, Tabriz University, Tabriz, Iran

Correspondence
Ashkan Safari and Mehran Sabahi, University of Tabriz, Tabriz, 51666-16471, Iran.
Email: ashkansafari.univ@gmail.com and sabahi@tabrizu.ac.ir

Abstract
In this paper, a simple and practical method to hookup between Pyboard and computer using MicroPython and MATLAB is presented. With the presented way, MATLAB can connect to Pyboard with virtual serial port (VSP). This process is performed with a virtual port, without using MATLAB toolbox in all versions of this software and control prototyping is widely available on the hardware. This system can also be used in Simulink and widely be under the control of MATLAB to perform tasks. The system is based on (.py) file and (.m) file. One is made in MicroPython to perform analog to digital task and the second contains VSP source code to have a virtual connection with the proposed board and calculating codes to plot graphics. This way can cause the high speed of data sampling and data transfer in two different environments: Python interpretation environment and MATLAB environment. With the defined way, it is possible to make the devices that require calculation operations and the correlation of the computer and external environment with lower costs and fewer accessories. To validate the correctness of the proposed approach, an experimental prototype as a total harmonic distortion (THD) meter device has been built.

1 | INTRODUCTION

Recent wide range of microcontroller applications have determined their advances in commerce and industry. Moreover, MATLAB is the common software used in this major. Accordingly, the connection between MATLAB and microcontrollers is the significant part of this process, performed with MATLAB toolbox. However, the quality of this tool can differ in different versions of the software. As a result, it can affect the time of the process. MicroPython STM32F411RE microcontroller (Pyboard) is one of the newly established boards that has noticeable capacities and runs on the entire rewrite of Python3.4 that called ‘MicroPython’. MATLAB/Simulink software uses toolboxes for targets such as graphical modelling to be performed on boards and even, this way performs the connection between the microcontroller and MATLAB can take time in data transfer and running Simulink on the proposed microcontroller [1]. The process of data stimulation was performed with Simulink and its block diagrams; however, this process done with MATLAB connection toolbox can affect the quality of the transferring data between the software and microcontroller [2]. Real-time control prototyping in microcontrollers with MATLAB/Simulink that is performed with the specific tools and specific stimulation blocks in MATLAB/Simulink that can differ in various versions of MATLAB [3]. Moreover, MATLAB graphical user interface (GUI) has the capability to control through MATLAB programming. This process is done with MATLAB/Simulink toolbox to have control on the microcontroller; However, this can have effects in time length of data transfer between MATLAB and microcontroller. Microcontrollers include interfaces that mostly used in electrical and computer projects. These interfaces can be under control with MATLAB/Simulink; however, since the type of the connection affects the quality of data communication, plays the main role and MATLAB toolbox is used in common boards [4]. The approach used in recent studies that are about the relation between MATLAB, MATLAB GUI and microcontrollers to

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2021 The Authors. IET Circuits, Devices & Systems published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology.
make connections, is MATLAB toolbox [5–8]. However, the
goodness of this tool can differ in various versions of MATLAB.

Here, a connection performed with a virtual serial port
(VSP) that is almost compatible with all versions of MATLAB.
Moreover, this technique creates a correlation between two
different environments (MicroPython and MATLAB) and
maximises the sampling speed, due to the RAM of Pyboard.
Accordingly, this way creates an optimum environment to
perform tasks, for example, GUI, data transfer, microcon-
troller connections, cross-platform, web-platform and console
applications. Therefore, it can be used in internet of things
(IoT), commerce, education and industry. To show the capa-
city of this method, a total harmonic distortion (THD) meter
device is implemented as a test bed, whereas this method could
be extended to run several industrial applications too.

2 | PYBOARD ARCHITECTURE

The MicroPython Pyboard is a microcontroller that runs
MicroPython, performing a low-level Python 3 operating sys-
tem that can be used to control electronic projects. Figure 1
depicts the overall schematic of a THD meter device based on
Pyboard. Accordingly, this board consists of 168 MHz Cortex
M4 CPU with hardware floating point, 1024KiB flash ROM
and 192KiB RAM, micro USB connector, for power and serial
communication. Micro SD card slot, supporting standard and
high capacity SD cards, 3× 12-bit analog to digital (ADC)
converters, available on 16 pins, 4 analog ground shielding,
2× 12-bit digital to analog converters, that are devised on pins
X5 and X6. It includes an on-board 3.3 V LDO voltage
regulator that can supply up to 250 mA, moreover, input
voltage range 3.6–16 V and DFU boot loader in ROM that
used for firmware easy upgrading.

3 | PROPOSED FRAMEWORK

This program is based on MicroPython framework that is
designed to be used on microcontrollers, including MicroPython
official boards and Arduino new series and so forth. Micro-
Python is an efficient derivation of Python 3 series programming
language and created with Python and C99. It includes a small
portion of Python standard library and optimised to run on
microcontrollers and in constrained environments. Micro-
Python is packed full of advanced features such as an interactive
prompt, precision integers, list comprehension, generators,
exception handling and so forth. It is compact enough to run
within 256k of code space and 16k of RAM.

4 | APPLICATION

4.1 | The concept of THD and DPF

THD and distortion power factor (DPF) are qualitative indexes
that illustrate how close the waveform is to the sinusoidal one.

\[
\text{THD} = \sqrt{\sum_{k=2}^{\infty} \left(\frac{I_k}{I_1}\right)^2} = \sqrt{\frac{I_2^2 - I_4^2}{I_1^2} - 1} \quad (1)
\]

\[
\text{DPF} = \frac{I_1}{I} = \frac{1}{\sqrt{1 + \left(\frac{I_2}{I_1}\right)^2 + \text{THD}^2}} \quad (2)
\]
Here, $I_1$, $I_{dc}$, $I_k$, and $I_{rms}$ are the rms values of the main harmonic, of the $k$th harmonic order, total rms value and average values of the load current, respectively. The rms value of the load current is calculated as:

$$I = \sqrt{\frac{1}{T} \int_{t_0}^{t_0+T} i(t)^2 dt}$$  \hspace{1cm} (3)$$

where $\omega = 2\pi f$ in which $f = 50$ Hz and $a_1$, $b_1$ are as follows:

$$a_1 = \frac{2}{T} \int_{0}^{T} i(t) \sin(\omega t) dt$$  \hspace{1cm} (4)$$

$$b_1 = \frac{2}{T} \int_{0}^{T} i(t) \cos(\omega t) dt$$  \hspace{1cm} (5)$$

Therefore, $I_1$ is calculated as:

$$I_1 = \sqrt{\frac{1}{T} \int_{t_0}^{t_0+T} [a_1 \sin(\omega t) + b_1 \cos(\omega t)]^2 dt}$$  \hspace{1cm} (6)$$

Also, the load active power and dc value of the load current are calculated as:

$$P = \frac{1}{T} \int_{0}^{T} i(t) \cdot v(t) dt$$  \hspace{1cm} (7)$$

$$a_0 = \frac{1}{T} \int_{0}^{T} i(t) dt$$  \hspace{1cm} (8)$$

5 | EXPERIMENTAL RESULTS

5.1 | System circuit

First, a THD test bed circuit has been made as shown in Figure 2, the circuit consists of a non-linear rectified inductive load with the addition of a half-wave rectifier to make a non-sinusoidal and asymmetric waveform of $i(t)$, and the current measurement section. The AC and DC components of the load current are applied to pin Y12 and pin X8 of the Pyboard, respectively. The last part of the measurement section connects to the Pyboard to send data and analysed with MATLAB. The AC and DC components of the load current are applied to pin Y12 and pin X8 of the Pyboard, respectively. Moreover, this process requires code (Experiment- B. MicroPython Code), for Pyboard that previously coded in its ‘main.py’ file. Then, Pyboard connects to the MATLAB with virtual USB (Experiment- C. MATLAB Connection Codes), and the coded program starts to sample data and save it in ‘THD data.txt’ file, then the file is opened with MATLAB, and the samples are analysed.

5.2 | MicroPython section

Depending on Figure 3, in Section 5.2, current $I$ is received from circuit with MicroPython codes written on board and pin Y12. The Pyboard side program that is written to sample the data of the load current and transferring to the computer is presented as:
# Imports
```python
import pyb
import time
from pyb import Pin, ADC
from pyb import USB_VCP
```

# Constants
```python
index=0
count=4000
duration=5000000
dat=[4000]
DC_Value=4000
```

# Setting
```python
pin1 = pyb.ADC(pyb.Pin.board.Y12) # create an ADC on pin Y12
pin2 = pyb.ADC(pyb.Pin.board.X8) # create an ADC on pin X8
vs = pyb.USB_VCP()
```

# Main Prog
```python
while (1):
    # Wait for Computer
    index=0
    while (index==None or index==0):
        index=vs.read()
        pyb.delay(200)
        pyb.LED(4).on()
        pyb.LED(4).off()
    # Data to Array "dat"
    pyb.LED(3).on()
    tStart = pyb.micros()
    i=1
    while (i<=count):
        dat.append(pin1.read())
        i=i+1
    tEnd = pyb.micros()
    duration=tEnd-tStart
    LSB=count&255
    MSB=count>>8
    vs.send(MSB)
    vs.send(LSB)
    pyb.LED(2).on()
    pyb.LED(4).on()
    BYTE0=duration&255
    BYTE1=(duration>>8)&255
    BYTE2=(duration>>16)&255
    BYTE3=(duration>>24)&255
    vs.send(BYTE3)
    vs.send(BYTE2)
    vs.send(BYTE1)
    vs.send(BYTE0)
    # DC Value transferring
    DC_Value=pin2.read()
    LSB=DC_Value&255
    MSB=DC_Value>>8
    vs.send(MSB)
    vs.send(LSB)
    pyb.delay(200)
    pyb.LED(3).off()
```

# Main Data transferring
```python
i=1
while(i<=count):
    LSB=(dat[i])&255
    MSB=(dat[i])>>8
    vs.send(MSB)
    vs.send(LSB)
    pyb.delay(200)
    pyb.LED(2).off()
```

According to the code and Figure 5, Pyboard samples the data of \( i(t) \) and converts it to a digital data with ADC, thus sends to MATLAB with VSP.
5.3  |  MATLAB section

Depending on Figure 4 and the illustrated results, the complete code that performs this process is represented below in two code blocks. The first block is shown below:

```matlab
%************ Setup
close all; % close all figures
clear all; % clear all workspace variables
clc; % clear the command line
fclose('all'); % close all open files
delete(instrfindall); % Reset Com Port
delete(timerfindall); % Delete Timers
START=1;
dat=[10000 10000];
INPUTBUFFER = 10000;
Pyboard = serial('COM5');
set(Pyboard,'InputBufferSize',INPUTBUFFER);
% ******************************************
while
    fopen(Pyboard);
```

```matlab
pause(0.2)
fwrite(Pyboard,START(l));
MSB=fread(Pyboard,1);
LSB=fread(Pyboard,1);
count=LSB+256*MSB
if count>10000
    count=10000;
end
BYTE3=fread(Pyboard,1);
BYTE2=fread(Pyboard,1);
BYTE1=fread(Pyboard,1);
BYTE0=fread(Pyboard,1);
duration=BYTE0+256*(BYTE1+256*(BYTE2 +256*BYTE3));
duration % [ uSec ]
SamplingTime=duration/count

MSB=fread(Pyboard,1);
LSB=fread(Pyboard,1);
DC_Value=LSB+256*MSB

for i=1:count
    MSB=fread(Pyboard,1);
    LSB=fread(Pyboard,1);
    dat(i)=LSB+256*MSB;
end

fid=fopen('THDdata.txt','w');
fprintf(fid,'%d
',count);
fprintf(fid,'%d
',duration);
fprintf(fid,'%d
',DC_Value);
for i=1:count
    fprintf(fid,'%d
',dat(i));
end
fclose(fid);
fclose(Pyboard);
pause
end
```

COM port can be different in other computers. After the data transferred, MATLAB performs the codes on the data and illustrates the result depending upon the related equations by the following program. The second code block that performs the equations on the data, is shown below:

```matlab
clc
clear
freq=50; % [ Hz ]
dat0=[1 1];
dat1=[1 1];
dat2=[1 1];
cal1=(0.185*27000/4700)/(2^12)*3.3; % Calibration coefficient of the used circuit
cal2=2.255e-4; % Calibration coefficient for
```
DC Value
offset=2^11; % Offset value for 12BIT ADC
we=2*3.14159265*freq;
Vm=12*(2^0.5);

%******** Data from file 'THDdata.txt' into 'dat0'

fid=fopen('THDdata.txt','r');
count=str2num(fgetl(fid));
duration=str2num(fgetl(fid)); % [uSec]
delta_t=duration/count/1000000;

% [Sec]
for i=head:count
    j=j+1;
    dat1(j)=dat0(i);
end

%******** Data selection from the zero origin of time
% (Zero point sync) and data into 'dat2'
sync=0;
for i=1:used
    if
        ((dat1(i)<=offset)&(dat1(i+1)>offset)&(sync==0))
            sync=1;
            ti=0;
            j=0;
        end
    if
        ((sync==1)&(ti<5/freq))
            if
                ((ti<=4.8/freq)||(ti>4.8/freq)&(dat1(i+1)<=offset))
                    j=j+1;
                    dat2(j)=dat1(i);
                    ti=ti+delta_t; % ti in [Sec]
                end
            end
    end

%******** Elimination about 30% of primary transient % part of dat0, next into 'dat1'
used=uint16(2*count/3);
head=count-used;
j=0;

for i=head:count
    j=j+1;
    dat1(j)=dat0(i);
end

%******** Calculation of RMS
% Irmsp=0;
for i=1:j
  Irmsp=Irmsp+(dat2(i)-offset)^2;
end
Irmsp=cal1 *((Irmsp/j)^0.5);
Irms=(Irmsp^2+Idc^2)^0.5

%******** Calculation of THD and DPF
P0=0;
I1=0;
I2=0;
I3=0;
I5=0;
ti=0;
for i=1:j
  P0=P0+sin(we*ti)*(dat2(i)-
  offset+Idc_offset);
  I1=I1+sin(we*ti)*(dat2(i)-
  offset+Idc_offset);
  I2=I2+sin(2*we*ti)*(dat2(i)-
  offset+Idc_offset);
  I3=I3+sin(3*we*ti)*(dat2(i)-
  offset+Idc_offset);
  I5=I5+sin(5*we*ti)*(dat2(i)-
  offset+Idc_offset);
  ti=ti+delta_t; % ti in [ Sec ]
end
Pin=Vm*cal1*P0/j
I1=2*cal1*(I1/j)/(2^0.5)
I2=2*cal1*(I2/j)/(2^0.5)
I3=2*cal1*(I3/j)/(2^0.5)
I5=2*cal1*(I5/j)/(2^0.5)

THD=((Irms^2-Idc^2)/I1^2-1)^0.5
DPF=I1/Irms
plot(dat2)
grid

According to the code and the proposed algorithm, the program gets the data from the txt file and performs the equations on the data and plots the related graphs.

5.4 | The results

The experimental prototype and waveforms of the load current are illustrated in Figures 5 and 6, respectively. From Figure 6, it is clear that load current is non-sinusoidal, data is analysed with the another MATLAB ‘.m’ file according to Equations (1)–(5) and the result is shown in Table 1.

6 | COMPARISON

In comparison to other ways, the represented approach has the capability to be used in all versions of MATLAB/ Simulink and besides MATLAB GUI, there is no requirement to have modification on the communication protocol. Accordingly, the connection method does not vary and performs the connection between the software and microcontroller to transfer the data, the additional information is shown in Table 2.

7 | CONCLUSION

Here, MATLAB software and a VSP have been used to develop an optimum environment, for the Pyboard microcontroller. A MicroPython code used to set up Pyboard, for

| Item | Description | Value |
|------|-------------|-------|
| THD  | Total harmonic distortion | 26.05 (%) |
| DPF  | Distortion power factor | 87.82 (%) |
| Idc  | DC component of the load current | 0.443 (A) |
| I    | Load current (rms) | 1.054 (A) |
| I1   | Main harmonic (rms) | 0.926 (A) |
| I2   | 2th harmonic order (rms) | 0.038 (A) |
| I3   | 3th harmonic order (rms) | 0.152 (A) |
| I5   | 5th harmonic order (rms) | 0.062 (A) |
| Pin  | Approximate active power load | 11.11 (W) |

| Item | Proposed approach | Other approaches |
|------|-------------------|------------------|
| Cross-platform capability | High | Low |
| Used microcontroller | MicroPython Pyboard | Common boards |
| Protocol stability in different projects | Stable | Unstable |
| Advances of proposed microcontroller | High | Low |
| Operating environment | Python and MATLAB | C and MATLAB |
| Programming language | MicroPython | C |
| Flexibility in various OSs | High | Low |
ADC data transfer and MATLAB code, for creating a virtual port, calculation part and the result plots. Due to the transfer time between Pyboard and MATLAB/Simulink, the virtual port plays the main role in the process.

As an experiment, the system used for a THD circuit to plot the THD graph.

In order, the results validate the possibility and correctness of this method. The experimental results, compared with the indicated specific measurement devices, validate the correctness of MicroPython microcontroller programming, Pyboard microcontroller, MATLAB codes and algorithms used for the study. With the presented way, it is possible to make the devices that require calculation operations and the correlation of the computer and external environment with lower costs and fewer accessories. The innovation of the study point can be summarised in five aspects:

(1) With the technique presented in the study, it is not required to modify the data transfer protocol.
(2) This approach even supports the GUI and Simulink without any inefficiencies of data transfer protocol.
(3) Using VSP to have a communication between computer software and microcontrollers.
(4) Using a capable microcontroller board called ‘Pyboard’ and an advanced microcontroller programming language called ‘MicroPython’.
(5) Creating an advanced microcontroller program, also cross-platform programs with the least lines of codes with the derivations of Python programming language, in comparison to other languages, especially C programming language, for microcontrollers.

ORCID
Ashkan Safari https://orcid.org/0000-0002-1780-7615

REFERENCES
1. Rusu, A., et al.: Rapid control prototyping toolbox for renesas m32e87 microcontroller. In: 2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), 1, pp. 1–6. IEEE, New York (2010). https://doi.ieeecomputersociety.org/10.1109/AQTR.2010.5520002
2. Rusu, C., Dadalescu, M., Balan, H.: Embedded toolbox for F24x dsk target microcontroller. In: 2007 International Aegean Conference on Electrical Machines and Power Electronics, pp. 556–559. IEEE, New York (2007). https://doi.org/10.1109/ACEMP.2007.4510564
3. Grepl, R.: Real-time control prototyping in matlab/simulink: review of tools for research and education in mechatronics. In: 2011 IEEE International Conference on Mechatronics, 881–886. IEEE, New York (2011). https://doi.org/10.1109/ICMECH.2011.5971238
4. Ivanov, V., Ivanov, S., Brjoujin, M.: Matlab graphical user interface for system with dallas microcontroller. In: 2008 International Symposium on Power Electronics, Electrical Drives, Automation and Motion, pp. 502–507. IEEE, New York (2008). https://doi.org/10.1109/SPEEDHAM.2008.4581196
5. Carlos, G.J., et al.: Designing of control using matlab and arduino for an electrohydraulic system from seismic simulation. In: 2017 IEEE XXIV international conference on electronics, electrical engineering and computing (INTERCON), pp. 1–4. IEEE, New York (2017). https://doi.org/10.1109/INTERCON.2017.8079720
6. Neaca, A., Neaca’a, M.I.: Matlab-simulink programming for the automated control of a resistive furnace. In: 2014 International Conference on Applied and Theoretical Electricity (ICATE), pp. 1–6. IEEE, New York (2014). https://doi.org/10.1109/ICATE.2014.6972605
7. Panda, A., et al.: Matlab data acquisition and control toolbox for basic stamp microcontrollers. In: Proceedings of the 45th IEEE Conference on Decision and Control, pp. 3918–3925. IEEE, New York (2006). https://doi.org/10.1109/CDC.2006.377158
8. Farooq, U., et al.: A low cost microcontroller implementation of fuzzy logic based hurdle avoidance controller for a mobile robot. In: 2010 3rd International Conference on Computer Science and Information Technology, 9, pp. 480–485. IEEE, New York (2010). https://doi.org/10.1109/ICCSIT.2010.5565096

How to cite this article: Safari, A., Sabahi, M.: Practical data connection between MATLAB and microcontrollers using virtual serial port and MicroPython Pyboard: A survey. IET Circuits Devices Syst. 15(5), 485–492 (2021). https://doi.org/10.1049/eds2.12038