International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 13 (2017) pp. 3944-3949
© Research India Publications. http://www.ripublication.com

Using of Inputs and Outputs on Microcontrollers Raspberry and Arduino

Michal Sustek, Miroslav Marcanik and Zdenek Urednicek

Faculty of Applied Informatics, Tomas Bata University in Zlin, Zlin, Czech Republic.

Abstract

Nowadays technology enables using of small devices to control complex systems. These devices (microcontrollers) can provide enough performance to substitute personal computers (PC). The two most known microcontrollers (Arduino and Raspberry) are connected together to one system. The article provides insight into the issue of inputs and outputs and their control for educational purpose.

Keywords: Microcontroller; Arduino; Raspberry; Input; Output; Python; General Purpose Inputs and Outputs

INTRODUCTION

Today's high demand for centralized and wireless control of a high number of devices [1]. That leads to simplification of our lives in home automation, laboratory project, education or entertainment. Microcontrollers are universal devices, which can provide enough of performance to control these types of projects. At the same time, the performance and capabilities of microcontrollers are growing fast in time [1].

Current industrial and professional systems for control inputs and outputs of a wide variety of component is based on specialized devices [2]. However, microcontrollers can realize these tasks. The microcontrollers are complex devices; moreover, one of them can be used in different tasks (for instance home automation, control system) [3,4].

There is a wide variety of manufacturers, which makes microcontrollers with a number of different types of processors and performances. The Raspberry and Arduino are the most common microcontrollers [1].

Section 1 describes two types of microcontrollers (Raspberry and Arduino). In section 2 is solved connection between these two microcontrollers. Section 3 shows basic information about inputs, and outputs and examples in programming by Python. Section 4 is focused on analog outputs, inputs, and conversion between them.

MICROCONTROLLERS

The microcontroller is a concept, which contains from Central Processing Unit (CPU), memory and inputs/outputs. The term microcontroller became popular after the introduction of a minicomputer when Isaac Asimov used term microcomputer in his story The Dying Night in 1956. [1]

The microcontroller replaced many components of minicomputer by a one-board device. From this time microcontrollers has gone through an amount of improvement and minimization until today's form of the microcontroller.

A. *Raspberry Pi*

The microcontroller Raspberry Pi is a small one-board microcontroller, which was focused on teaching a basic computer science in schools. In particular, Python programming language. This small microcontroller became more popular than anticipated.

Raspberry evolved through several versions with different performance. Today's generation of microcontroller, Raspberry PI 3 is equipped with Broadcom BCM2837 SoC with 1.2 GHz quad-core ARM Cortex-A53 processor. This processor is ten times more powerful than a processor, which was used, in the first version of Raspberry. It supports Linux-based operating systems, USB ports for keyboards, mouse, Ethernet adapter and many other devices. On the board, it is HDMI connector for attaching monitor and GPIO (General Purpose Input/Output) pins. As a data storage are used microSD card and primary programming language is Python.

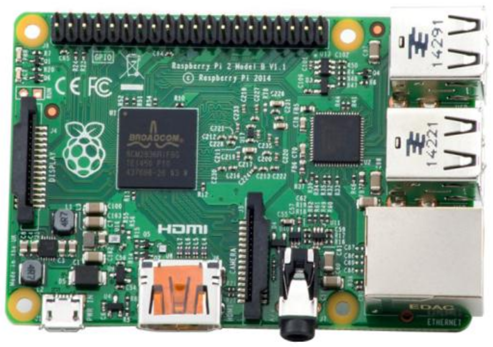


Figure 1: Raspberry Pi 2B [13].

For this project was chosen variant Raspberry Pi 2B, which has 4-core 900 MHz processor, 1 GB of RAM, 4 USB 2.0 ports, HDMI video output, 40-pin GPIO header.

B. Arduino

The microcontroller Arduino is another one-board microcontroller. Its design served for support to teaching informatics and automation in schools. Arduino does not support the connection of monitor, keyboard, and mouse. On the other hand, it is designed to the connection of LED, LCD displays, sensors, actuators, and lighting.

Arduino is designed in a wide variety of types, like Raspberry. Unlike Raspberry, Arduino is not designed for substitute a standard PC. The main program is developed on another device or PC and it is uploaded into Arduino. Only this program is launched on Arduino and it contains loops, which are constantly repeated. Arduino can be expanded by Arduino Shields. These modules can improve Arduino's performance (Wi-Fi, Ethernet, Motor, GPS). [5]

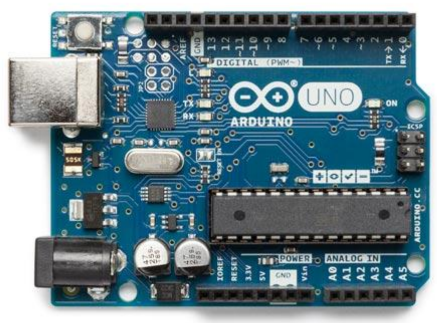


Figure 2: Arduino UNO [12].

CONNECTION BETWEEN MICROCONTROLERS

These two different microcontrollers can work together to enhance their capabilities. There are some examples why use Arduino and Raspberry together.

- Using a large number of shareable examples and libraries for Arduino
- Working with 5 V logic levels (Raspberry has 3.3 V logic)
- Enhancing Arduino experiments with more processing power.

Arduino development board must be programmed by the external device, like PC or Raspberry. Debug will be faster if the Raspberry is used to programming, on the other hand, the compilation will be slower. The Arduino IDE must be installed on microcontroller Raspberry, which can be done by terminal and commands:

```
Sudo apt-get update
Sudo apt-get install Arduino
```

Now the Arduino must be plugged into one of the open USB serial port. This connection will be able to provide enough power for its running. However, in several tasks is better to use external power source (running motors or heaters for instance).

When the Arduino IDE has launched it pools all the USB devices and builds a list that is shown in the Serial Port menu. [4] In this list, where the port is and in the board menu must be selected type of Arduino board (for instance Uno). Then the python script must be created and launched.

The Python script:

```
import serial


port_A =”/dev/tty/ACM0”
ArduinoSerial = Serial.Serial (port_A,9600)
ArduinoSerial.flushInput()
while true:
    if (ArduinoSerial,inWaiting()>0:
            input = ArduinoSerial.read(1)
            print(ord(input))
```

In program is first imported library for working with a serial port, and then is opened serial port, which is connected to Arduino. In next part of the script, the input buffer is cleared out, from the serial buffer is read one byte. Finally, is changed the incoming byte into an actual number with *ord()*.

Then the Arduino send a number to the script. The number is interpreted as a string. This solution is sufficient if just one byte is sent (for instance 1 for left button, 2 for right button) – it can be used 255 values for discrete events. For using a range 0 to 1023, the Arduino code must be modified by the following code.

```
void setup() {{
Serial.begin(9600);
}
void loop () {
        for (int b=0; b < 1024; b++)
        Serial.println (b,DEC);
        delay(50)
        }
}
```

The main difference between these two codes is in *println()* command. In first case *serial.write()* is used to write a raw number to the serial port. On the other hand, using of *println()* lead to format a number as a decimal string and send the ASCII codes for the string [6].

On the Python's side can be used command *readline()* instead of *read()*, which will read all of the characters up until the carriage return and newline. In Python, exist a wide variety of functions for converting between the various data types and strings. For instance can be used command *inputAsInteger=int (input)*, which change the input to the integer value.

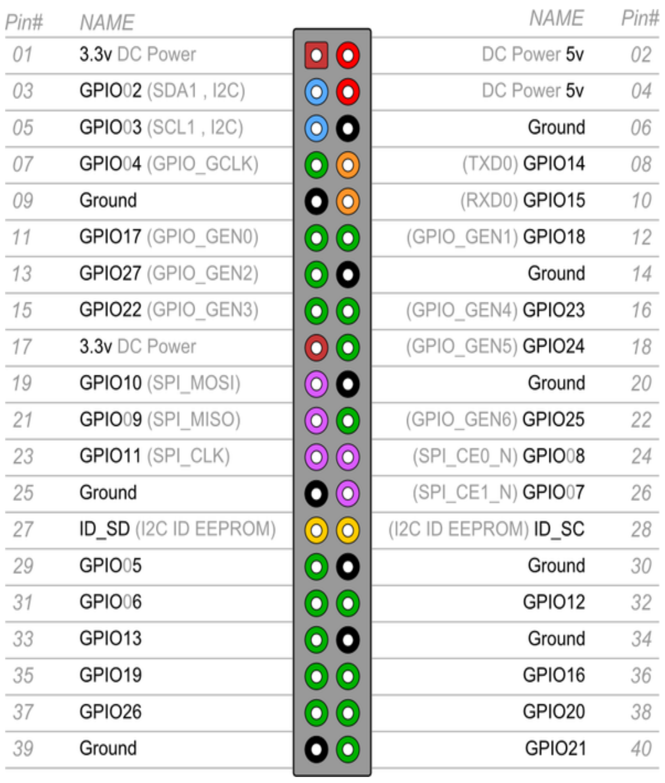
INPUTS AND OUTPUTS

The main advantage of Raspberry is using GPIO pins. They are generic pins on Raspberry board, which behavior is not defined. Their purpose can be programmed to users' needs. All pins can be configured as inputs or outputs.

Properties:

- Input values can be readable
- Output values can be readable or writable
- Can be used as IRQ (Maskable interrupt)

In the case of raspberry are GPIO pins controlled by external library RPi.GPIO, which must be imported into the main control program. This library can be downloaded from www.python.org.



Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I2C)	DC Power 5v	04
05	GPIO03 (SCL1 , I2C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I2C ID EEPROM)	(I2C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Figure 3: GPIO layout Raspberry Pi 2 B [14].

GPIO pins can be accessed for controlling many hardware equipment like LED, motors, relays. That are examples of outputs. As input can be used button status, switch status or dials, and sensors output, which means light, motion, temperature or proximity. Raspberry Pi has besides classical GPIO also some inputs and outputs for usage of keyboard, monitor, mouse, Ethernet. It provides crucial advantage together with Arduino microcontroller.

There are also few more advantages for using Raspberry:

- File system – ability to read and write data in Linux system
- Linux tools – usage of command-line utilities to control processes and automate wide variety of tasks
- Languages – wide variety of supported programming languages (for instance Python, C, Java, Perl)

A. Programming

In this chapter will be presented elemental programming of basic inputs and outputs (for instance LED as an output, and button as an input). These two examples are digital input and output.

1) LED blinking

```
import RPI.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(25, GPIO.OUT)

while True:
    GPIO.output(25, GPIO.HIGH)
    time.sleep(1)
    GPIO.output(25,GPIO.LOW)
    time.sleep(1)
```

First, the libraries for GPIO control and sleep function are imported. Then the GPIO is set as BCM (the chip signal numbers are used) and pin 25 is set as output. Last part of the program consists of infinite loop in which are LEDs on/off function and waiting function for 1 second.

2) *Button reading*

```
import RPI.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(24, GPIO.IN)

counter=0

while True:
        Value_input=GPIO.input(24)
        if (Value_input==True)
                Counter=counter+1
                Print ("Button was pressed: " +str(counter)
                "times")
        time.sleep(0.2)
```

As in the previous code, it is necessary to import libraries for GPIO and sleep functions. The GPIO is set as BCM and port 24 is set as an input. Then the variable counter was created. Last part of the program also consists of infinite loop, in which is controlled that the button is pressed. Then counter is incremented and the message with a number of pressing is showed.

ANALOG INPUT AND OUTPUT

In many application is necessary to work with the continuous signal (for instance dim of LED, control of motor speed). In this case is perforce to convert a digital signal into an analog value.

A. Output – digital to analog

GPIO module uses Pulse-width modulation method (PWM), which can be modified to seem like the range of voltage; however, it is pulsing of PWM signal as one can see in figure 4 [1]. The behavior of this signal for controlled device is similar to an analog signal. In programming, are used syntax *range*. For range are set start value, final value and step (for instance *range(0,100, 10)*).

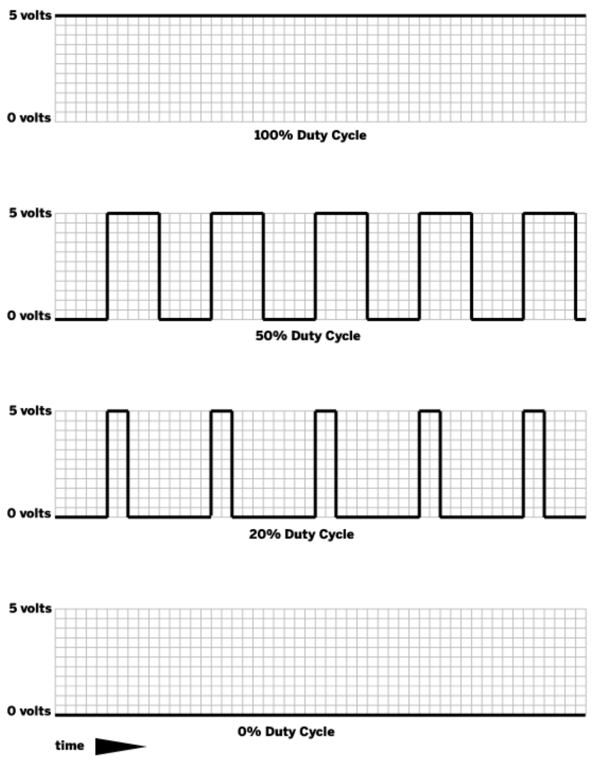


Figure 4: PWM range [1].

The following example is created for dimming of blinking LED.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(25, GPIO.OUT)
A=GPIO.PWM (25, 50)
A.start (0)

while True:
        for DC in range (0,100,10)
                A.ChangeDutyCycle(DC)
                time.sleep (0.1)
        for DC in range (100,0,-10)
                A.ChangeDutyCycle (DC)
                time.sleep(0.1)
```

Most of the program is similar to the program for on/off the LED. The different parts are the creation of PWM object A and set it to GPIO pin 25 with a frequency 50 Hz and start value 0. The second difference is part of the code for incrementing the value of DC by 10 from starting value to the value 100 (100 percent of duty cycle). Moreover, the reverse part of the program for decrementing DC value [7].

B. Input – analog to digital

Like can be outputs controlled on a scale 0 to 100% of the duty cycle, can be also possible to read sensors that offer to Raspberry Pi a range of values.

On microcontroller Arduino, there is a special hardware, which can convert the analog voltage value into the digital data (for instance ADS1115) [1]. This hardware is not available on Raspberry, but it is possible to use Arduino hardware because Adafruit create a special breakout board for it.

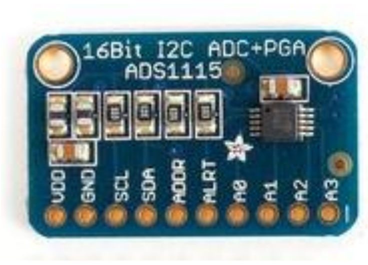


Figure 5: ADS1115 [7].

Adafruit also provides open source Python library for reading the values from the ADS1115, via I2C (Inter-Integrated Circuit) protocol. To read analog inputs via I2C must be I2C enable on Raspberry configuration, and drivers for ADS and tools for I2C must be installed [1].

The following program code is created for writing the code to read the ADC.

```
from Adafruit_ADS1x15
from time import sleep
ADC=Adafruit_ADS1x15.ADS1115()

while True:
        output1=ADC.read_ADC()
        print output1
        sleep(0.25)
```

The program contains creation of new ADS object called ADC. Addition reading from channel A0 on the module and store this value into variable output1.

CONCLUSION

The high demand for home automation and Do It Yourself (DIY) project leads to an extension of microcontrollers in a wide spectrum. This paper provides insight into two main microcontrollers on the market and their basic programming their inputs and outputs in Python language.

These two main microcontrollers are Raspberry and Arduino. Raspberry is more similar to the personal computer, and it can substitute it. On the other hand, Arduino is used more in home automation like a device, which is programming on an external device that leads to usage into the control system in real time. In the project was used a combination of Raspberry and Arduino. As a programming language was chosen Python, which is a higher programming language. Python does not need strictly defined variables, unlike C++. Next to Arduino was used GPIO pins, so the module RPi.GPIO must be imported into programs. In project were created programs for controlling outputs and work with inputs, and their analog alternatives. These programs were focused on education.

In future, it is going to expand programs into more complex variant for the whole system and creating of educational aids for microcontroller programming.

ACKNOWLEDGMENT

This work was supported by Internal Grant Agency of Tomas Bata University under the project No. IGA/FAI/2017/04.

REFERENCES

[1] RICHARDSON, Matt, and Shawn P. WALLACE. *M ake: getting started with Raspberry Pi*. Sebastopol, CA: OReilly, 2015.

[2] MOLLOY, Derek. *Exploring Raspberry Pi: interfacing to the real world with embedded Linux*. Indianapolis, IN: John Wiley & Sons, 2016.

[3] WENTK, Richard. *Raspberry Pi*. Indianapolis, IN: John Wiley & Sons, Inc., 2014.

[4] SPANNER, Günter. *Arduino: circuits & projects guide*. Aachen: Elektor International Media b.v., 2013.

[5] KARVINEN, Kimmo, and Tero KARVINEN. *Getting started with sensors:*. Sebastopol, CA: Maker Media, 2014.

[6] BELL, Charles. *Beginning sensor networks with Arduino and Raspberry Pi:*. Berkeley, CA: Apress, 2013.

[7] BANZI, Massimo, and Michael SHILOH. *Getting Started with Arduino:*. Sebastopol, CA: Maker Media, 2015.

[8] CHAITANYA, K. et al. *Multiple home automation on raspberry pi*. . . V. BHATEJA, S. C. SATAPATHY a A. JOSHISpringer Verlag, 2017. Sponsors:; Conference code: 180309.

[9] MEMBREY, P., D. VEITCH a R. K. C. CHANG. *Time to measure the pi*. Association for Computing Machinery, 2016. 327-334 s. Sponsors: ACM SIGCOMM; ACM SIGMETRICS; Conference code: 124785. ISBN 9781-450345262(ISBN).

[10] NAVEENKRISHNA, M. a S. JAYANTHY. Real time vehicle tracking and monitoring using raspberry pi. *International Journal of Applied Engineering Research*. 2015, vol. 10, no. 20, s. 15259-15263.

[11] PATHIK, B. B. et al. *Development of a cell phone based vehicle remote control system*. , 2014.

[12] Arduino, product list, Available from: https://www.arduino.cc/en/main/products (2017-15-6)

[13] S. MONK. *Make: action: Movement, light, and sound with Arduino and Raspberry Pi.* San Francisco, CA: Maker Media. (2016)

[14] Raspberry Pi, GEEK; Available from: http://www.raspberry-pi-geek.com/howto/GPIO-Pinout-Rasp-Pi-1-Model-B-Rasp-Pi-2-Model-B (2017-12-6)