Microcontroller Raspberry Pi 2B as control system for basic types of motors

MICHAL ŠUSTEK, MIROSLAV MARCANÍK, ZDENĚK ÚŘEDNÍČEK
Department of Automation and Control Engineering
Tomas Bata University in Zlín
Nad Stráněmi 4511, 760 05 Zlín
CZECH REPUBLIC
sustek@fai.utb.cz, marcanik@fai.utb.cz, urednicek@fai.utb.cz

Abstract: - Microcontroller's technology is widely used in diverse field; including automation and issue of remote control of moving objects; that is caused by expanding capabilities of these devices. In this project, a microcontroller Raspberry Pi 2B was chosen for controlling basic types of motors (DC motors, servo-motors and stepper motors). This paper serves as insight into issue of programming motors on Raspberry Pi 2B by Python programming language and connection between microcontroller and other components on breadboard. This information can be used in education process or for those who want use a microcontroller Raspberry in their work.

Key-Words: - **Microcontroller, Raspberry, Python, DC motors, Servo motors, Stepper motors, Control system, Remote control**

1 Introduction

Wi-Fi, GSM or Bluetooth controlled devices are very popular today. These technologies provides a simplification of our lives in the field of home automation or entertainment (mainly in form of radio-controlled models) [1]. At the same time, the performance and possibilities of microcontrollers have risen in many applications and areas [2].

Most of current control systems are based on specialized devices, which provide controlling of specific object. On the other hand, the small attention is given to universal devices (like microcontrollers). Microcontrollers are able to perform a wide variety of tasks in home automation, movement control and more [1, 2]. The remote control of object by microcontrollers is popular among researchers and "RC fans" than among public or industry [3].

Today's microcontroller can manage a complex applications, because their performance growth in last years. There are also wide variety of manufacturers which created microcontrollers with diverse types of processors and performance. Among to best known and used microcontrollers are Raspberry and Arduino [4]. Both of them provide high performance, and they can be used in many challenging automation applications.

The second generation of the Raspberry microcontroller provides enough performance to replace a standard PC in some audiovisual and automation applications [5]. With the use of the wireless extension, which can be achieved by a simple antenna, the device becomes in a complex control station.

Section 1 describes a microcontroller Raspberry Pi and its performance. Section 2 shows a basic functionality of DC motors, servo-motors and stepper motors. In section 3 is insight into issue of connection between components. In addition, section 4 describes basic programming in language Python for one component of each type (1 motor, 1 servo-motor, 1 stepper motor).

2 Microcontroller Raspberry

A Raspberry Pi 2B is a small single-board device, which supports Linux-based operating systems, USB connections for mouse, keyboard, Ethernet adapter, and other devices. On Raspberry board are HDMI connector for attaching a monitor and general-purpose inputs and outputs [6]. A MicroSD card is used a storage device and Python is used for programming.

In history, Raspberry Pi was created in UK as a low cost platform for teaching computer basics, in particular Python [8]. The first generation of Raspberry was introduced in February 2012. Since this time, a wide variety of Raspberry microcontrollers has been created. In addition, each

of these variants have different performance and parameters.

Raspberry Pi 2B was chosen for the purpose of this research. It contains 4-core 900 MHz processor, 1 GB of RAM, 4 U SB 2.0 por ts, HDMI video output, 40-pin GPIO (General Purpose Input/Output) header.

Fig. 1 Microcontroller Raspberry Pi 2B [8].

Raspberry can be used as an unpretentious and cheap replacement of a PC; however, it has wide use in home automation and remote control systems. This is caused by an opportunity to use GPIO pins, which function is programmable.

2.1 GPIO pins

General-Purpose Input/Output are generic pins on integrated circuit of Raspberry. Behavior of these pins is not defined and they can be programmed according to users' needs. Each pin can be configured as input or output; it can be enabled and disabled; input value can be readable and output value can be readable or writable [8]. In some applications are GPIO pins used as maskable interrupt (IRQ).

Ability of using the GPIO pins is provided by external Python module RPi.GPIO. This module must be imported into the main control program.

3 DC motors and servo-motors

DC motors and servo-motors are main actuators part in each robotic system. These components provide movement and rotation around desired axis. In deeper context, this component provide ability of movement. DC motors and servo-motors are crucial in all robotic projects. [12]

3.1 DC motors

Direct current motors are composed of three main parts (rotor, stator, and commutator). The stator circumferentially is provided with regularly spaced and mutually oppositely oriented main magnetic poles and commutation poles. The poles of the same polarity follow the poles of the given polarity in the direction of rotation of the anchor (rotor). The rotor has coils in the grooves and these coils are connected to a mechanical commutator. The commutator provides the supply of a correctly oriented current to the coils of the rotating anchor so that all currents of the flowing coil side form a torque of the same direction in the magnetic field of the main poles [12].

On magnetic neutral place between main poles of commutator, are placed carbon brushes. The number of the brushes is the same as the number of the main poles.

Current, which flows through anchor windings, creates reactionary magnetic field that deforms and weakens magnetic field around main poles and has effect on commutator magnetic field. A compensating winding is used to suppress the reactionary magnetic field. Today a stator and a rotor are created from isolated dynamo-sheet of metal in modern motors.

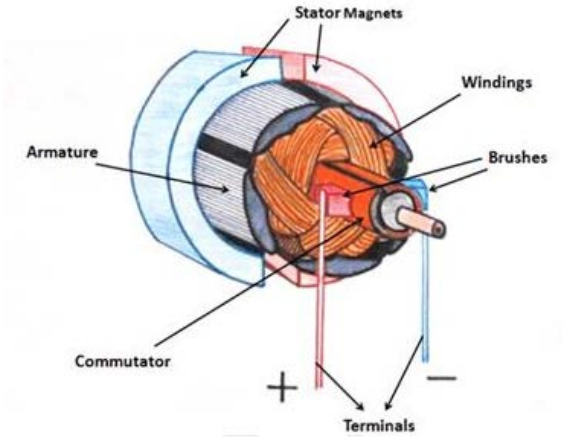

Fig. 2 Brushed DC motor [14].

Brushless DC motors are the second variant of DC motors. Brushless motors provide electrical commutation with permanent magnet rotor and stator with a sequence of coils. A permanent magnet rotates and current carrying conductors are fixed in this type of motor. Transistors or rectifiers at the correct rotor position switch the armature coils in such a way that the armature field is in space quadrature with the rotor fields [12].

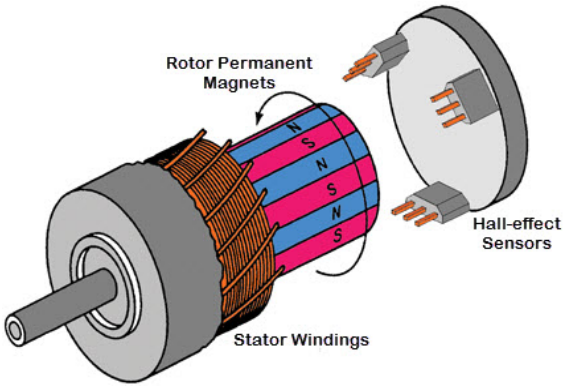

Fig. 3 Brushless DC motor [15].

3.2 Servo-motors

A servo-motor (servo) is a motor which uses a feedback to correct the output of the motor. The feedback is based on information from a sensor or from a sensors and external circuitry. The servo itself is consisted from a DC motor, a potentiometer and a control circuit. They are small but very energy-efficient devices, which are controlled by electrical impulses. Gears for controlling a shaft attach the motor. The power supply of the motor is stopped, when the motor shaft is at the desired position. If it is not in desired position, then it is turned in the appropriate direction. The motor speed is proportional to the difference between desired and actual position [12].

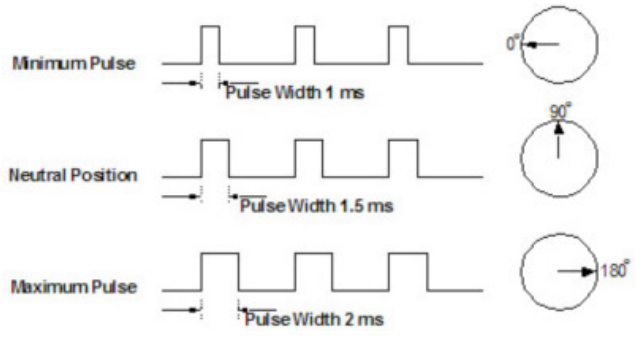

Fig. 4 Servo-motors controlling pulses [8].

Signals are send as electrical pulses with variable width, or a Pulse Width Modulation (PWM) is used. A minimal and a maximal value of pulse and repetition rate is usually distinguished. Usually servos can turn 90 in each direction for total 180□.

The neutral position is defined as the position, where the motor have amount of potential rotation equal to direction in both side (clockwise and counter-clockwise). The servo expects a signal every 20 milliseconds and the length of the electrical pulse determines how far the motor turns. When the servo moves on position, it will hold that position. If any external force try to push against the servo, while it holding a position, the servo will resist from moving out of the position. The maximum amount of force which a servo can resist is called the torque rating. The position will not be held forever, the position pulse must be repeated to instruct servo to stay in desired position.

There are two types of servos [13], AC and DC. AC servos can work with higher current surges and are used in industrial applications. On the other hand, DC servos are not designed for high current surges and are better usable for smaller applications. There are also servos for continuous rotation.

In real application servos are used in RC models (airplanes, walking robots), service robots and operating grippers.

3.3 Stepper motors

Stepper motors are DC motors, which movement is in discrete steps. These motors can convert a train of input pulses into precisely defined increment in the shaft position. Stepper motors have usually multiple toothed electromagnets, which are around central gear-shaped piece of iron. An external circuit or microcontroller energizes these electromagnets [8].

The principle of stepper motor is based on electromagnets and magnetic attraction. To one magnet is given power and the gear's teeth are attracted to the electromagnet. In next step, the first electromagnet is set off and second electromagnet is set on. The process is then repeated. In that way can be the motor turned by a precise angle.

There can be distinguished three types of stepper motors:

- Permanent magnet stepper
- Hybrid synchronous stepper
- Variable reluctant stepper

Permanent magnet stepper motor (PM) use permanent magnet in the rotor part and the attraction or repulsion is controlled by stator's electromagnet. PM motors are called canstack rotor.

Variable reluctant motors have iron rotors and they are based on principle that the minimum reluctante occurs minimum gap. So the rotor points are attracted toward the stator magnetic poles.

Hybrid motors contains from permanent magnet similar to PM stepper motors. In addition hybrid motors are axially magnetized (one end is polarized to north; the second end is polarized to south). Stator and rotor assemblies in these types of motors have tiitg-like projections, which are align in various configuration during rotation [19].

Also can be used two basic winding arrangements for electromagnetic coils in a two-phase stepper motor.

- Bipolar
- Unipolar

Bipolar motors uses H-bridge circuitry to reverse the current flow through the phases. Energizing of the phases alternate the polarity. That leads to the fact that all the coils can be put to work turning the motor.

Unipolar motors energize the phases always in same way. That means one lead will be always positive and the other will be always negative. This type of motors can be implemented by simple transistor circuit, but the disadvantage is that there is less available torque, because on one time can be energized only half of the coils [19].

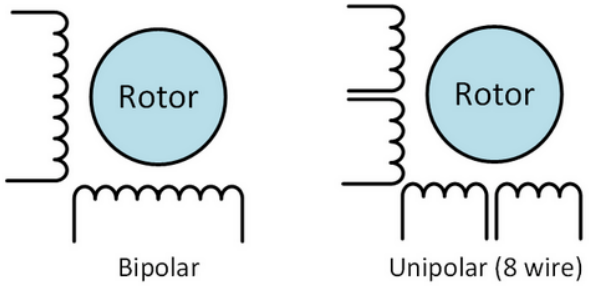

Fig. 5 Principle of stepper motor [8].

The main advantages of stepper motors are [8]:

- Low speed control
- Speed control
- Positioning

On the other hand, the limitations are [8]:

- No feedback
- Limited high speed torque
- Low efficiency

4 Motor connection

A small 6V DC motor PERMAX 280, a microcontroller Raspberry Pi 2B, and an H-bridge L293D were used in this project. The H-bridge is a simple circuit, which contains switching elements. These elements are usually Field-Effect Transistors (FET) transistors. All switches can be turned off or on independently. The H-bridge can be used to switch a direction of a motor depending on a current flow. Table 1. presents how switch state can effect the behavior of a motor.

Tab. 1 H-bridge switch combination [8].

S-1	S-2	S-3	S-4	Motor behavior
1	0	0	1	Clockwise turns
0	1	1	0	Counter-clockwise turns
0	0	0	0	Stop
1	1	-	-	Short circuit
-	-	1	1	Short circuit
1	0	1	0	Braking
0	1	0	1	Braking

Where 1 means the switch is closed, 0 means the switch is opened, and – means it does not matter on the state of a switch.

The H-bridge L293D contains 2 H-bridges, so it is useful for 2 DC motors. The main advantages of this chip are:

- Thermal protection
- Capable with Raspberry logic (3V)
- Voltage range of 4.5 to 36V for motors
- Motor's peak current of 1.2A
- Continuous motor current of 600 mA

In this work, all the components are connected on a breadboard as depicted in Figure 5. The main advantage of using L293D with Raspberry (it has 3V logic) is that the control of pins need only a little current to control motors.

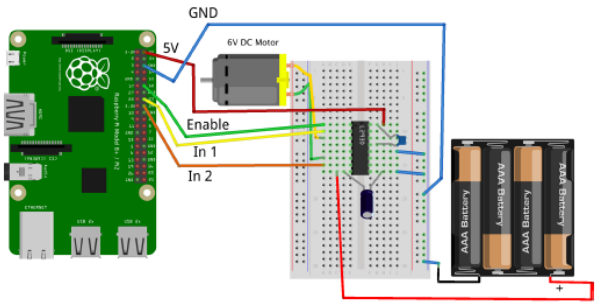

Fig. 6 DC motor wiring with Raspberry Pi 2B [8].

5 Software solution

Python has been chosen as the programming language in this work. Python is a higher programming language, which does not need to have strictly defined variables, unlike C++. In the deeper context, it is a hybrid dynamically interpreted

language from a group of scripting languages. Graphic User Interface (GUI) IDLE, which is well organized, was used to develop the software itself. A module called Rpi.GPIO has been used together with basic libraries.

5.1 Solution for DC motor

At the beginning, there is a need to import a module for controlling GPIO pins, in particular RPi.GPIO, which can be downloaded from www.python.org.

```
import RPi.GPIO as GPIO
import time
```

Then the mode for GPIO pins must be set up. BOARD and BCM setup can be chosen. BOARD option specifies referring of the pins to the plug. BCM option means referring of the pins by the Broadcom SOC channel.

```
GPIO.setmode(GPIO.BCM)
```

There is also a need to select GPIO pins, which will be used. The H-bridge L293D is used as a motor driver. Therefore, pin number 18 must be enabled to control the speed of the motor.

```
enable_pin = 18
pin_1 = 23
pin_2 = 2
```

All used pins will be configured as outputs. These pins help to control direction of the motor. We also define PWM analog output, where 500 is PWM frequency. The initial value of duty cycle is set to 0% of the frequency.

```
GPIO.setup(enable_pin, GPIO.OUT)
GPIO.setup(pin_1, GPIO.OUT)
GPIO.setup(pin_2, GPIO.OUT)
pwm_motor=GPIO.PWM(enable_pin,500)
pwm_motor.start(0)
```

In the following block of code, the movement function for each direction (forward, backward) and stop function are defined. Pin 1 is responsible for forward movement, on the other hand pin 2 is used for reverse movement.

```
def forward(duty_cycle):
   GPIO.output(pin_1, True)
   GPIO.output(pin_2, False)
   motor_pwm.ChangeDutyCycle(duty_cycle)

def reverse(duty_cycle):
   GPIO.output(pin_1, False)
   GPIO.output(pin_2, True)
   motor_pwm.ChangeDutyCycle(duty_cycle)

def stop():
   GPIO.output(in_1_pin, False)
   GPIO.output(in_2_pin, False)
   motor_pwm.ChangeDutyCycle(0)
```

The following part of the program represents the main loop which prompts the user for a command and calls direction functions and the stop function.

```
try:
   while True:
      direction = raw_input('w – forward, x – reverse, t - stop')
      if direction[0]=='t':
         stop()
         else:
            duty_cycle= input('Duty cycle (0-100%)')
            if direction [0]=='w':
               forward(duty_cycle)
            elif direction [0]=='s':
               reverse(duty_cycle)

finally:
   print("Cleaning up")
   GPIO.cleanup()
```

This is elementary program for control DC motor with the microcontroller Raspberry Pi 2B. This program is written for one motor, which is wired to the microcontroller by L293D motor driver.

5.2 Solution for servo motor

The module RPi.GPIO must be used for GPIO control as in the previous case.

```
import RPi.GPIO as GPIO
import time
```

The number of GPIO pin on Raspberry Pi 2B has to be chosen. In addition, every servo needs a slightly different length of pulse to maximize its range of angles, two constant are used to set the pulse duration between angles 0□ and 180□. Values of these variables represent duration of pulse in milliseconds. Frequency is set up to 50 Hz, therefore it is giving a pulse every 20 milliseconds.

```
servo_pin=18
0_deg = 0.5
```

```
180_deg = 2.5
frequency= 50.0
```

Some calculations, which are related to the length of pulse, were made for easier future modifications. Period is 1000 milliseconds and is divided by frequency (50 Hz in our case) so the result is 20 millisecond. In addition, if there is a need to change a duty cycle, an interval between 0 and 100 must be used and the constant k can be used to scale an angle to duty value. To convert the pulse for 0□ to a corresponding value of duty between 0 and 100, is length of pulse multiplied by constant k. The value of range of duty is calculated by multiplying the span of pulse length by the constant k.

```
period = 1000/frequency
k = 100 / period
0_deg_duty = 0_deg*k
pulse_range = 180_deg-0_deg
duty_range=pulse_range*k
```

The part, which initialize the GPIO pin, is similar to the variant for DC control by Raspberry Pi 2B.

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(servo_pin,GPIO.OUT)
pwm=GPIO.PWM(servo_pin, frequency)
pwm.start(0)
```

In angle definition, we convert angle into duty cycle value, and then we call ChangeDutyCycle to set the new pulse length.

```
def set_angle(angle):
   duty_cycle=0_deg+(angle/180.0)*duty_range
   pwm.ChangeDutyCycle(duty_cycle)
```

The main loop of control program is written below. In the following part, a value of angle between 0□ and 180□ is set. It could be used for steering, when 90 is forward direction, 0□ is turning left and 180□ is turning right.

```
try:
   while True:
      angle = input("Angle 0 □ to 180□")
      set_angle(angle)

finally:
   print("Cleaning up")
   GPIO.cleanup()
```

The program serves as a basic control software for servomotor by microcontroller Raspberry Pi 2B. Program is written for one servo-motor

5.3 Solution for stepper motor

As in all previous cases, it is necessary to use modules RPi.GPIO and time. In addition, the mode for GPIO pins must be set up.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
```

In this part of program are defined constans for four GPIO pins and set them to be outputs. These pins use pins number 23, 24, 25, 18. Period is set to 0,02 second which corresponds to frequency 50 Hz.

```
pin1_input = 23
pin2_input = 24
pin3_input = 25
pin4_input = 18

GPIO.setup(pin1_input,GPIO.OUT)
GPIO.setup(pin2_input,GPIO.OUT)
GPIO.setup(pin3_input,GPIO.OUT)
GPIO.setup(pin4_input,GPIO.OUT)

period=0.02
```

The part which define forward and reverse steps for 4 coil stepper motor. The four coil activations take place in the appropriate order using coil_setup function. This steps are repeated with a delay of period between each coil's change.

```
def forward_step (steps, period):
        for i in range (0,steps):
                coil_setup (1,0,0,1)
                time.sleep(period)
coil_setup (1,0,1,0)
                time.sleep(period)
                coil_setup (0,1,1,0)
                time.sleep(period)
coil_setup (0,1,0,1)
                time.sleep(period)

def reverse_step (steps, period):
        for i in range (0,steps):
                coil_setup (0,1,0,1)
                time.sleep(period)
coil_setup (0,1,1,0)
                time.sleep(period)
```

```
                coil_setup (1,0,1,0)
                time.sleep(period)
coil_setup (1,0,0,1)
                time.sleep(period)

def coil_setup (in1,in2,in3,in4):
        GPIO.output(pin1_input,in1)
        GPIO.output(pin2_input,in2)
        GPIO.output(pin3_input,in3)
        GPIO.output(pin4_input,in4)
```

The main control loop of program reads the command string using raw_input. The parameter that follows the letter is first cut off the command using the syntax [1:]. That means, in Python, the string from position 1 to the end of the string. Then is the parameter converted into number by using the int built-in function. The last part of main control loop is series of three if statements, then carry out the appropriate action for the command.

```
try:
   print('command letter followed by number');
   print('p20 – set the inter-step period to 20 ms (control speed');
   print('f100 – forward 100 steps');
   print('r100 – reverse 100 steps');

while True:
   command = raw_input (,Enter command:')
   parameter_str = command [1:]
   parameter = int(parameter_str)
   if command [0]=='p':
      period = parameter/1000.0
   elif command [0]=='f':
      forward_step(parameter,period)
   elif command [0]=='r'
      reverse_step(parameter,period)

finally:
   print('Cleaning up')
   GPIO.cleanup()
```

This is simple program for control of stepper motor by microcontroller Raspberry Pi 2B. Used stepper motor is a bipolar.

The stepper motor's rotation is not very smooth. For most application it is not a problem, however, in some specific application the smoother movement is needed. It can be used PWM to energize coils more smoothly, that leads to smoother rotation of the motor. This system is called microstep. In general is easier to use hardware, which is specifically designed to microstep of stepper motors (for instance the EasyDriver board, based on the A3967 IC). The following program is created for microstep of stepper motor on microcontroller Raspberry Pi 2 B and EasyDrive Board.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode (GPIO.BCM)

pin_step=24
pin_direction = 25
pin_ms1 = 23
pin_ms2 = 18

GPIO.setup(pin_step, GPIO.OUT)
GPIO.setup(pin_direction, GPIO.OUT)
GPIO.setup(pin_ms1, GPIO.OUT)
GPIO.setup(pin_ms2, GPIO.OUT)

period=0.02
```

The previous part of program is very similar to control program for stepper motors. It must be imported specific Python libraries, GPIO mode must be set and each used pin must be defined.

```
def step(steps, direction, period):
   GPIO.output (pin_direction, direction)
   for i in range(0, steps):
      GPIO.output(pin_step, True)
      Time.sleep(0.00002)
      GPIO.output(pin_step, False)
      Time.sleep(0.00002)
```

Step funcion contains code for moving the motor on by a number of steps or microsteps[]. Used parameters are direction and period. Firstly the direction pin is set on the EasyDriver board and then are generated the required number of pulses on the step pin.The lenght of pulse is 2 microsecond.

```
def step_mode(mode)
   GPIO.output(pin_ms1, mode & 1)
   GPIO.output(pin_ms2, mode & 2)
```

Step mode function sets the stepping mode pina to value of mode. This value can be chosen between 0 and 3. In this syntaxe is separated 2 bits of the mode number and sets 2 variables using the logical and operator.

```
try:
   print('Command leter followed by number');
```

```
print('p20 – set the inter-step period to 20 ms (control speed)');
print(m- set stepping mode (0 – none, 1- half, 2 – quarter, 3 – eight)';
print ('f100 – forward 100 steps');
print ('r100 – reverse 100 steps');

while True:
    command = raw_input ('enter command:')
    parameter_str = command [1:]
    parameter = int (parameter_str)
    if command [0]=='p':
      Period=parameter/1000.0
    elif command [0] == 'm'
      Step_mode(parameter)
    elif command [0] == 'f'
      Step(parameter, True, period)
    elif command [0] == 'r'
      Step(parameter, False, period)
finally:
    Print ('Cleaning up')
    GPIO.cleanup()
```

The main control loop of the program is very similar to previous program for control of stepper motors. To original code is added m command, which allows set of the microsteps.

6 Conclusion

Microcontrollers can be used in wide variety of task, mostly thanks to increase of performance and high demand for universal wireless devices that lead to significant use of microcontrollers. The paper provide elementary insight into the issue of controlling individual types of motors by microcontroller (in particular a microcontroller Raspberry Pi 2B).

This type of microcontroller provides enough performance to run control system for DC motors, servo-motors or stepper motors. 6V DC motor PERMAX 280, motor driver L293D and breadboard have been chosen for creation of this system. In second variation was used servo motor Turnigy™ TG9e Eco Micro Servo and in third part was used stepper motor SX17-1005LQCEF. Connection between components is experimental and it is not used in real robotic platform yet. As a main programming language is used Python; however, there is also needing to use GPIO pins, so the module RPi.GPIO must be imported into the program. The submitted syntaxes in programs are the elementary way to control DC motors, servo-motors and stepper motors.

The next work will be focused on extending the control system on a cell phone as a control device, in particular, by using LTE (Long Term Evolution) technology. The main difference in LTE system will be in control software (Android, iOS application). An LTE modem must be added to the entire system. Another further work can be aimed at implementation of this control system into a 4-wheeled robotic chassis.

Acknowledgment

This work was supported by Internal Grant Agency of Tomas Bata University under the project No. IGA/FAI/2017/004.

References:

[1] Names of the Authors, Title of the Paper, *International Journal of Science and Technology*, Vol.X, No.X, 200X, pp. XXX-XXX.

[2] Names of the Authors, *Title of the Book*, Publishing House, 200X.

[3] A. A. Asadi, S. Bagheri, A. Imam, E. Jalayeri, W. Kinsner, and N. Sepehri. Institute of Electrical and Electronics Engineers Inc. (2016)

[4] K. Chaitanya, G. Karudaiyar, C. Deepak, S. B. Reddy, 1st International Conference on D ata Engineering and Communication Technology, 469 Springer Verlag. (2017)

[5] I. Lobachev, E. Cretu Institute of Electrical and Electronics Engineers Inc. (2016)

[6] A.C. Martinez, International Conference on Human Factors and System Interactions, 497 Springer Verlag. (2016)

[7] P. Membrey, D. Veitch, R. K. C. Chang. Association for Computing Machinery, (2016)

[8] M. Šustek, M. Opluštil, Z. Úředníček, 6th International Masaryk Conference for Ph.D students and young researchers, (2015)

[9] M. Vanitha., M. Selvalakshmi, R. Selvarasu. Institute of Electrical and Electronics Engineers Inc. (2016).

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

[11] H.L. Dai, L. Wang. Communications in Nonlinear Science and Numerical Simulation 46: 116-125 (2016)

[12] K. Premkumar, K. G. J. Nigel. "Smart Phone Based Robotic Arm Control using Raspberry Pi, Android and Wi-Fi."Institute of Electrical and Electronics Engineers Inc. (2015)

[13] D. Sanchez-Benitez, J. M. de la Cruz, G. Pajares, D. Gu. "Visual Control of a Remote Vehicle." Intelligent Robotics and Applications, Pt Ii 7102: 579-588 (2011)
[14] M. Ghosh, S. Ghosh, P.K. Saha, G.K. Panda, IEEE Transactions on Industrial Electronics, 64 (2017)
[15] R.S. Ashok, Y.B. Shtessel, American Control Conference (ACC) (2016)
[16] Todd H. Hubing, Clemson University [online], Available from: http://www.cvel.clemson.edu/auto/actuators/images/chaudhary-DCmotor.png (2017.5.18)
[17] Electrical Technology, UK, Birmingham, Available from: http://www.electricaltechnology.org/2016/05/bldc-brushless-dc-motor-construction-working-principle.html (2017.5.18)
[18] M. Šustek, M. Marcaník, P. Tomášek and Z. Úředníček, DC Motors and Servo-motors Controlled by Raspberry Pi 2B, *21st International Conference on Circuits, Systems, Communications and Computers (2017)*
[19] B. Schwebber, "Stepper Motors Make the Right Moves with Precision, Ease and Smarter Drivers", Mouser Electronics, Available from: http://www.mouser.com/pdfdocs/PublicRelations_TechArticle_StepperMotors_RightMoves_2015Final.PDF?cm_mmc=PressRelease-PR-_-MOUSER-_-Stepper_Motors_Make_the_Right_Moves_with_Precision,_Ease_and_Smarter_Drivers-_-2015-02-16