STUDIA INFORMATICA 2016
Volume 37 Number 4A (127)

Paweł RYBKA, Krzysztof WOSIK, Łukasz SZCZEPAŃSKI
Silesian University of Technology, Institute of Electronics
Adam ZIĘBIŃSKI, Rafał CUPEK
Silesian University of Technology, Institute of Informatics
Roman WYŻGOLIK, Sebastian BUDZAN
Silesian University of Technology, Institute of Automatic Control

POWER MANAGEMENT AND SENSORS HANDLING ON THE AUTONOMOUS MOBILE

Summary. Day-to-day everywhere in the world grows a tendency automate various areas of our everyday life. For many recent years worldwide companies compete with each other in developing a conception of fully automated, autonomous car. Such idea is now being analyzed and investigated by scientists at universities and engineers from automotive companies. This article describes a part of AutoUniMo project concerning importance and usage of different sensor modules in design and construction of autonomous mobile platform based on the Raspberry Pi with Linux operating system. The project consists of following modules – accelerometer, gyroscope, magnetometer, wheel encoder and dedicated Raspberry Pi HAT (Hardware Attached on Top) for power management.

Keywords: Raspberry, sensors, accelerometer, gyroscope, magnetometer, encoder, power management

ZARZĄDZANIE ENERGIĄ I OBSŁUGA SENSORÓW W MOBILNEJ PLATFORMIE AUTONOMICZNEJ

Streszczenie. Każdego dnia zwiększa się w świecie tendencja do automatyzowania zagadnień dotyczących zajęć życia codziennego. W ostatnich latach światowe koncerny rywalizują na polu rozwiania koncepcji w pełni zautomatyzowanego samochodu. Idea ta jest ciągle rozwijania i analizowana przez naukowców każdego liczącego się uniwersytetu oraz inżynierów firm motoryzacyjnych. W artykule opisano część projektu AutoUniMo traktującą o znaczeniu i wykorzystaniu modułów sensorycznych w mobilnej platformie autonomicznej osadzonej na komputerze Raspberry Pi (Raspbian Linux). W skład projektu wchodzą - akcelerometr, żyroskop,

magnetometr, enkoder dla kół oraz Raspberry Pi HAT zaprojektowany m.in. do zarządzania poborem energii.

Słowa kluczowe: Raspberry, sensory, akcelerometr, żyroskop, magnetometr, enkoder, zarządzanie energią

1. Introduction

Day-to-day everywhere in the world grows a tendency to automate everyday activity. For many recent years worldwide companies compete with each other in developing a conception of fully automated, driverless car. Such idea is now being analyzed and investigated by scientists and engineers in all respectable universities of technology and automotive companies [1, 2, 3].

Our project also concerns this problem and in this paper we focus only on sensor platform, especially on setting up connection between sensors and Raspberry Pi unit, reading values generated by sensors and processing them in such a way, that the gathered information could be used to model actions to be undertaken by the car.

There also arises another problem which needs a solution. For the platform which is powered by Lithium-Polymer battery packages operation time dramatically drops as number of modules powered by the platform rises [4, 5]. That's the reason we developed a PCB board – Raspberry Pi HAT, which allows the computer to turn the unused devices off (as even when not performing any action they still draw current from the batteries).

2. Raspberry Pi platform

The platform we have decided to use is the Raspberry Pi. Tis has been dedicated by the relatively low price while the performance is sufficient to meet the power consumption and data processing requirements in the project. It runs on Linux based OS – Raspbian, making it easily accessible and versatile. The board also allows networking which is crucial for data communication. Moreover the manufacturer provides Genarla Purpose Input/Output (GPIO) pins along with crucial for sensor connecting low-level peripherals:

- I^2C
- UART
- SPI

Another thing worth mentioning is that the board consumes around 700 mA which is very small for a platform of this computing power, which is essential for a mobile vehicle

operating on batteries. The platform is very popular in many applications, including these connected with vehicles [6, 7].

2.1. Power management

The platform draws current from Lithium-Polymer onboard packages which power Raspberry Pi together with all other sensors and engines. It is known, that the fewer current is drawn from the batteries – the longer the platform can operate on a single accumulator. Thus, power consumption management is crucial for remote applications like this, and also significantly reduces the costs of exploitation.

Sensors and devices, even if not used, draw current from the batteries. The idea of reducing power usage in this area is to create an electronic device which turns the devices off if they are currently unused and power them back on, when required. For this purpose there should be created a programmable key (its visual representation is shown in Fig. 1), which would allow controlling of current flow for each device that is included in the process.

Fig. 1. Visual representation of programmable key
Rys. 1. Wizualna reprezentacja programowalnego klucza

3. Sensor modules section

To give the platform ability to recognize its orientation and dynamic state [8], it was equipped in appropriate sensors, which are listed and shortly described below.

3.1. Magnetometer

Magnetometer which was used in project is GY-271 module, equipped in *Honeywell* HMC5883L chip. It is digital sensor capable of measuring magnetic field in 3 axis [9].

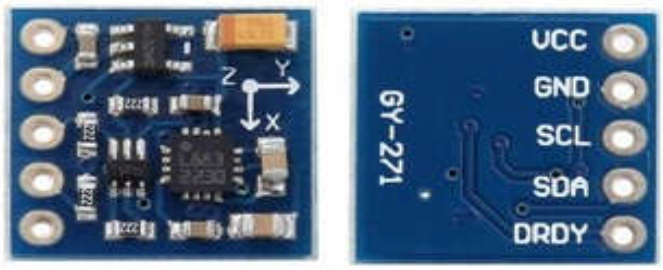

Fig. 2. Gy-271 module
Rys. 2. Moduł Gy-271

Sensor operates in voltage range 2.16 V - 3.6 V, thus it can be directly connected to Raspberry Pi. It has also low power consumption (100 mA while performing measurement, and only 2 mA when idle), which is crucial when mounting on battery driven mobile platform. Depending on the gain settings, maximum available accuracy is 0.73 mG, and maximum measurable intensity is 8 G. Maximum data rate, the device is able to collect measurements, is 160 Hz. However it requires additional connection and interrupt handling which is not supported by Raspbian (operating system managing Raspberry Pi). Sensor has also embedded self-test procedure, which is advised to be performed on every start-up, and (as it is the only way) it communicates with Raspberry Pi using I^2C bus.

3.2. Gyroscope

Gyroscope used in this project is PmodGYRO module with *STMicroelectronics* L3G4200D chip, also three axial, digital output device [10]. It is motion sensor manufactured in Micro Electro-Mechanical Systems (MEMS) technology. Device is able to measure angular velocity with 8.75 $m\frac{^\circ}{s}$ maximum precision, and 2000 $\frac{^\circ}{s}$ maximum range. It operates in voltage range 2.4 V-3.6 V, and during normal operation consumes 6.1 mA. However power consumption can be reduced when measurements are not taken, by putting device in sleep (1.5 mA) or power-down (5 mA) modes. Device possess two sets of communication protocols pins, which is SPI, and I^2C, but for universality of communication I^2C has been chosen. Sensor poses also few worth mentioning features, like embedded self-test procedure, it's stability over temperature range, temperature sensor(from which data are also collected), 32 slots FIFO queue for storing measurements, and high-pass and low-pass filters.

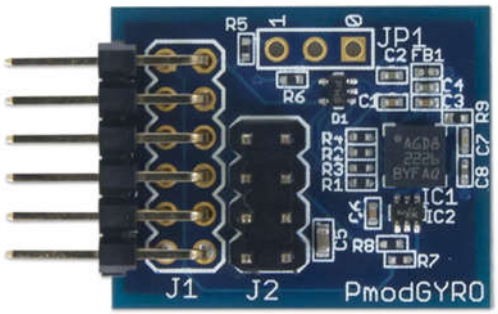

Fig. 3. PmodGyro module
Rys. 3. Moduł PmodGyro

3.3. Accelerometer

Module which was used in project is PmodACL from Digilent powered by the Analog Devices ADXL345 [11] acceleration sensor.

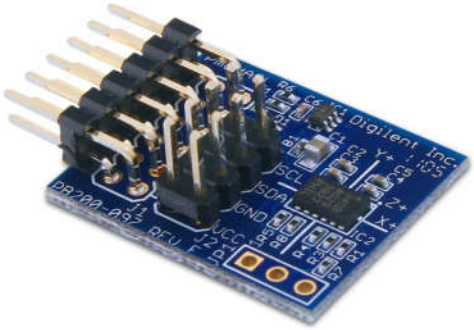

Fig. 4. :PmodACL module
Rys. 4. Moduł PmodACL

ADXL345 is a 3-axis digital accelerometer which allow to measure acceleration in range up to ±16 g with sensitivity range from 3.9 to 256 LSB/g (depend from selected output resolution and g-range) and can survive shock of 10 000 g.

This sensor come with mechanism that detect events like single/double tap, free fall etc that are reported by interrupts on specific pins of module.

It is ultralow power device. It operates in voltage range 2-3.6 V and consumes 23 µA during normal operation and 0.1 µA in standby mode.

Module was equipped in SPI and I^2C communication interfaces. In project PmodACL was connected to platform by I^2C interface.

3.4. Optical encoder

Encoder used in project was designed for DFRobot rovers installed on platform and it can be identified by Stock Keeping Unit (SKU) part number SEN0038.

Encoder uses non-contact method of measurement and it can give rotation degree of the wheels with resolution 20 pulses per revolution.

It operates in voltage range 3.3-5 V and consumes current below 20 mA. To communication it used one signal wire which inform about rotation by changing state on wire between HIGH and LOW during rotation of wheel.

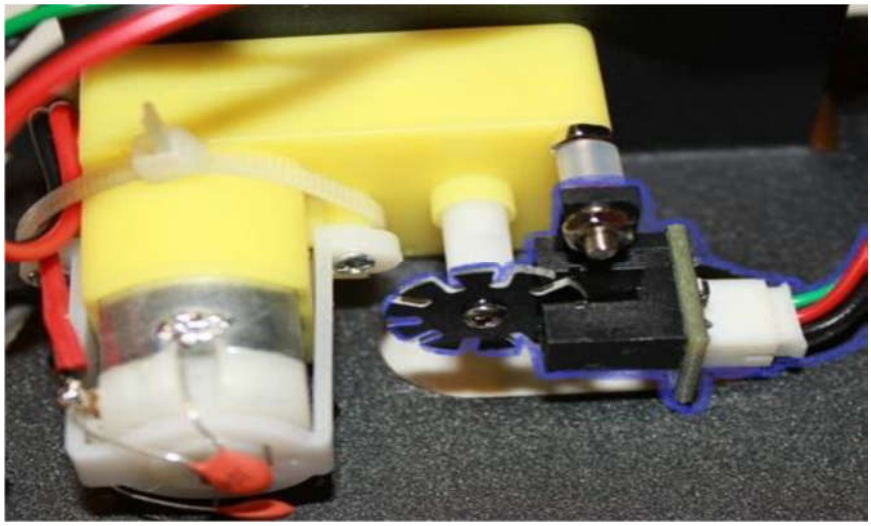

Fig. 5. Optical Encoder SEN0038 mounted on rover
Rys. 5. Enkoder optyczny SEN0038

4. Integration of sensors modules with the platform

4.1. Sensors

All sensors in project were handled by their own C++ library written for project purposes.

To meet project requirements, the libraries treat sensor like an objects with three main functions responsible for: initialize of sensor, making measurement and turn the sensor off.

This model of code was introduced to simplify references to sensors from main program of platform. All functions have the same name for all sensors but have owns algorithms.

4.1.1. Magnetometer

In the initialization method, besides the establishment of I^2C communication and setting configuration register, self-test procedure take place. Device generates artificial magnetic field of known magnitude, performs measurement, and saves results. Next, artificial magnetic field is removed, second measurement is performed, results are subtracted from previous ones, and saved in data output registers. After this operation in data output registers there should appear values of magnitude of the artificial magnetic field only, which is specified by the manufacturer, and allows to test proper functionality of the device.

Two calibration method has been implemented to improve usefulness of the measurements: hard and soft iron. Hard iron calibration cancels effect of constant magnetic field in proximity of sensor, like power supply wires(when current is approximately constant), or objects behaving like solid magnets. During this part of calibration, magnetometer collects data about magnetic field in XY plane. This allows it to determine net artificial magnetic field strength acting on it, which is next subtracted from each next measurement. Soft iron calibration cancels the effect of curving the magnetic field by magnetically non-neutral objects, and it is performed after hard iron calibration(when only distorted Earth's magnetic field is measured). Procedure measures magnetic field in XY plane, and calculates scale factor which is multiplied be every next X-axis measurement. This operation distorts X-axis raw magnetic field data, but improves accuracy of compass needle heading, which is the main task of magnetometer in this project.

After calibration measurements can be performed. Two modes of operation are available:

- Single measurement mode. In this mode, sensor takes only one measurement, returns results to the output data register which are being read, and saved in the class fields. After measurement, sensor goes into idle state, to reduce power consumption.
- Continuous measurement mode. Sensor performs measurements at maximum available rate (160 Hz), and returns results into data registers, which can be read at any moment. This mode increases power consumption, but assures that read values are the most recent.

When measurement has been performed, values read from sensor are being stored in the fields of the class. To access them one must call method that is responsible for returning results into data structure (see Table 1).

Table 1

Data structure for storin results of magnetometer measurement

Name	Type	Description
rawX	Float	Magnetic field strength measured along X axis
rawY	Float	Magnetic field strength measured along X axis
rawZ	Float	Magnetic field strength measured along X axis
rose	Float	Compass needle headings

In the data structure there have been returned values of measured magnetic field strength along each axis, and headings of compass needle, calculated using equation (1)

$$\text{rose}=\text{arctg}\left(\frac{\text{rawX}}{\text{rawY}}\right)\text{-}90° \quad (1)$$

90° shift is performed for convenience of the user, because intuitive "front" of the sensor(and whole platform) is shifted by this value.

What is also worth to notice that at current state of the project, soft iron calibration distorts returned raw results. This is irrelevant for most possible applications in which user will be interested only in compass needle heading(because only ratio between rawX and

rawY matter). When one is more interested in the raw magnetic field measurement, calibration should be omitted. This "issue" will be fixed in the nearest future.

For the needs of the project there also has been created way to visualize measurements coming from the sensors. Using TCP/IP protocol data are send from Raspberry Pi to the server application running on another machine in the network. Received data are being presented on the time plots. Figure 6 present magnetic field strength measured on X and Y axis, and compass needle headings calculated using equation (1), while platform was performing more than 360° turn.

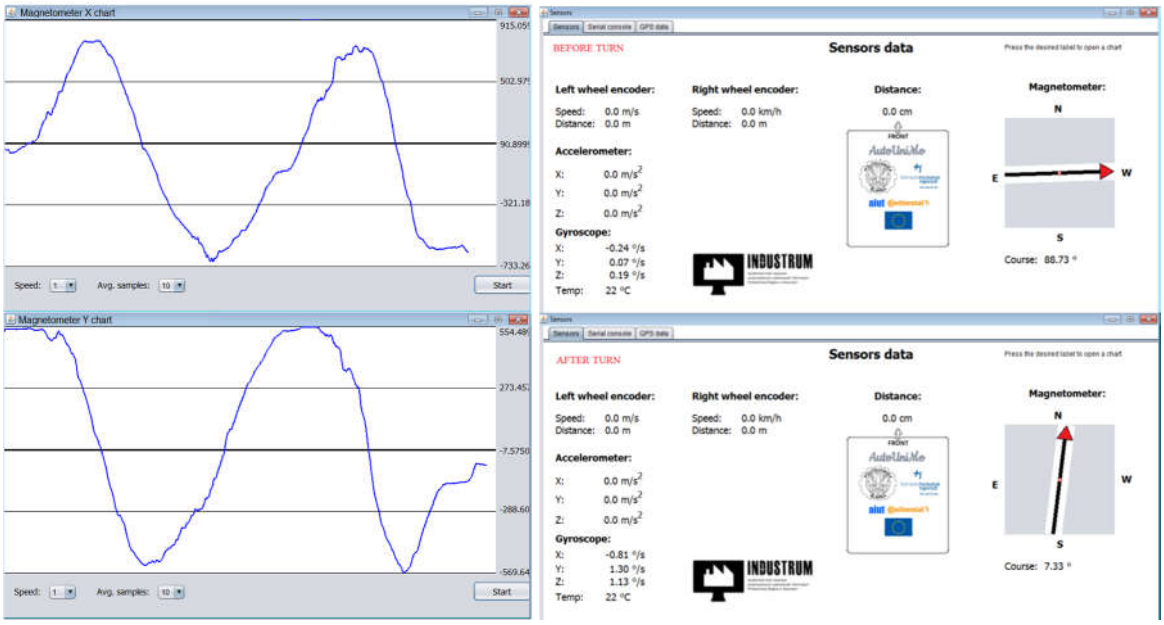

Fig. 6. Test of the magnetometer
Rys. 6. Test magnetometru

4.1.2. Gyroscope

Initialization method is responsible for setting all configuration registers, establishment I^2C communication, and (as in case of magnetometer) performing self-test procedure. Apart from technology details, device simulates rotation on each axis of known value, and place results in data output registers, which are being read. Measured values are then compared to the device manufacturer specification, and if they are in acceptable range, device is operational.

Calibration method applied to the gyroscope is much simpler than in case of magnetometer, because there is no such need. L3G4200D possess embedded flash memory in which it stores calibration coefficients, which are automatically downloaded to the registers after device power-up. The only calibration method which was implemented is finding zero level. It is done by averaging 100 measurements, when platform is still, which are next

subtracted from every subsequent measurement(for each axis). This simple procedure almost completely cancels steady state error.

After calibration measurements can be taken. Sensor by default operates in continuous measurement mode, and taking measurement only read the most recent value stored in the data output registers. It is also worth to mention about temperature sensor readings. Whereas the angular velocity readings only needs to be scaled, the temperature sensor returns values in non-intuitive way. For 25 °C the value of output temperature register is 25, and is changing by -1 for each °C increase. This means that to obtain temperature measurement, value from register must be subtracted from 50.

When measurement has been taken, one can obtain its result, by calling method that is responsible for returning them to the data structure, presented in Table 2.

Table 2

Data structure for storing results of gyroscope measurement

Name	Type	Description
Yaw	float	Angular velocity with respect to X axis
Pitch	float	Angular velocity with respect to Y axis
roll	float	Angular velocity with respect to Z axis
Temp	short	Actual temperature measured by the sensor

Using mentioned previously visualization application, time plot for gyroscope measurements have been obtained and are presented in the Fig. 7, and Fig. 8.

In the Fig. 7 one can observe change in X, Y, and Z axis while platform drives onto the slope, stops for a while, and next drives backwards. It can be seen that main response to that movement comes from Y axis. Other axis response can also be observed but it is relatively small, and probably comes from irregularity of the ground, which gyroscope can measure.

Fig. 8 presents measurement results while platform was performing 90° turn. Main response comes from Z axis, while other gives very small readings, which also can be interpreted as irregularities of the ground.

4.1.3. Accelerometer

Initialize function simply establish I^2C connection and parameters of sensor work, like measurement range, resolution of measurement or data acquisition mode.

All parameters which can be set in this function are listed in table 3.

In measurement function data from internal sensor registers was drawn and processed to obtain results from sequence of zeros and ones stored in registers.

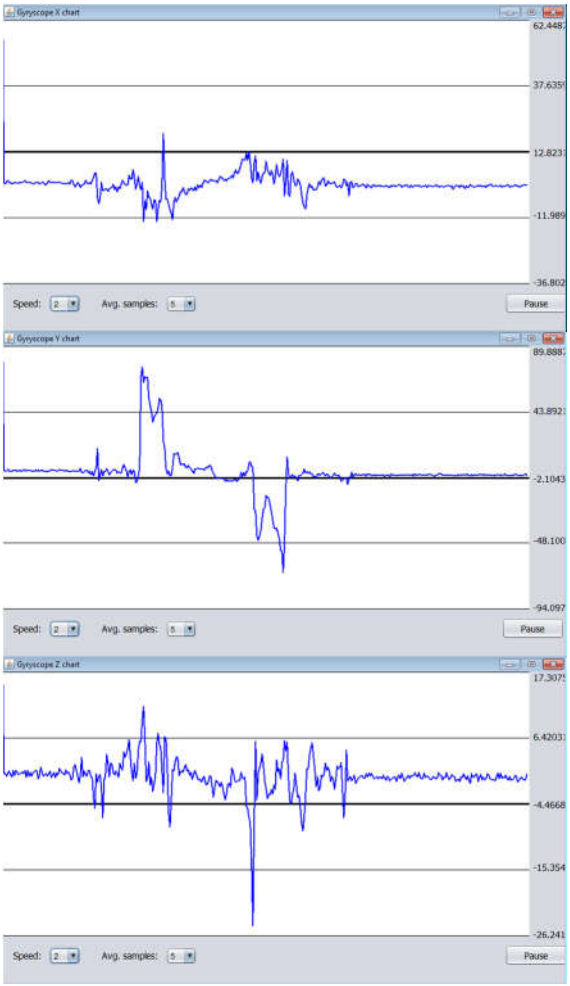

Fig. 7. Test of the gyroscope(turn)
Rys. 7. Test żyroskpu (turn)

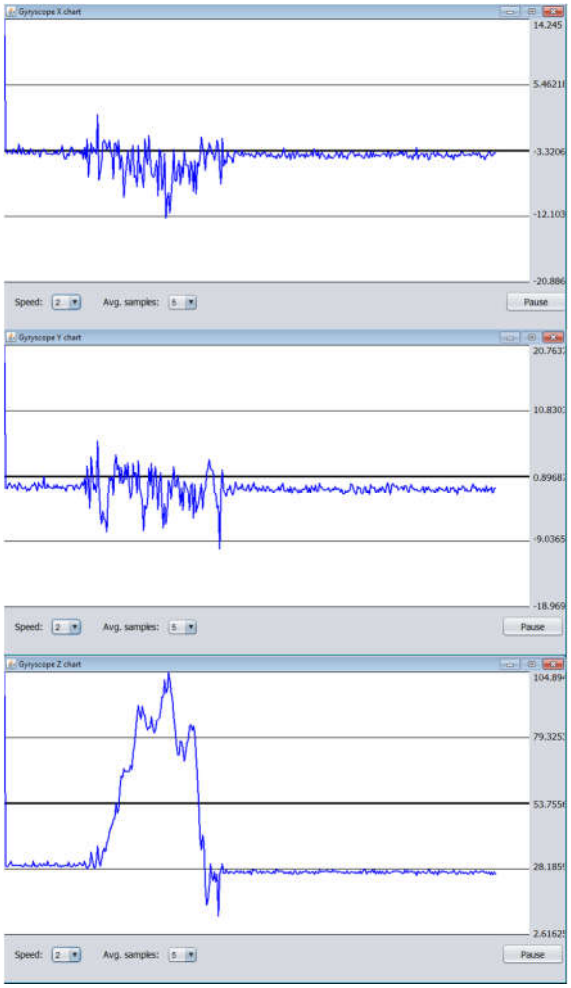

Fig. 8. Test of the gyroscope(slope)
Rys. 8. Test żyroskopu (slope)

Table 3

Settable parameters of accelerometer

Name	Available values
Bandwidth rate	25 Hz/50 Hz /100 Hz /200 Hz /400 Hz
Ranges of measurement	2G/4G/8G/16G
Resolution of measurement	ful_res - full resolution mode fix_res - fixed resolution mode
Mode of data acquisition	Bypass, fifo, stream, trigger
Address of sensor for I^2C protocol	Depend of sensor

To calculate acceleration a_G in g and m/s^2 units the following formulas was used:

$$a_G = \text{Sensor\_\#axis\_data} * \text{scale factor} \quad (2)$$

$$a_{m/s2} = a_G * Fg \quad (3)$$

Value of scale factor from formula (2) can be found in datasheet of sensor. It depends on parameters of sensor which was set in initialize function. Fg in formula (3) is gravity force of earth, Fg=9.8 (approximately).

Raw data form sensor and processed ones was stored in variables of specified type. Raw data was stored in 16 bit integer variables called raw_data.#. g and m/s^2 data was stored in float variables called g_data.# and ms2_data.# , where # is letter of axis.

Off function switch sensor to standby mode for reduce power consumption when usage of accelerometer was unwanted.

In Fig. 9, 10 and Fig. 11 the sensor work for all axis is presented. Results are good. Tested axis has value of acceleration close to gravitational force while other axes have small readings. This error occurs because mathematical formulas assume perfect model of world and of course because of disturbances caused by other sensors, motors of platform and noise from environment. The sensor does not give perfect values however all errors may be reduced, but it can be done only after connection of all other sensors to platform, because it is necessary to eliminate all disturbances that can occur in one time.

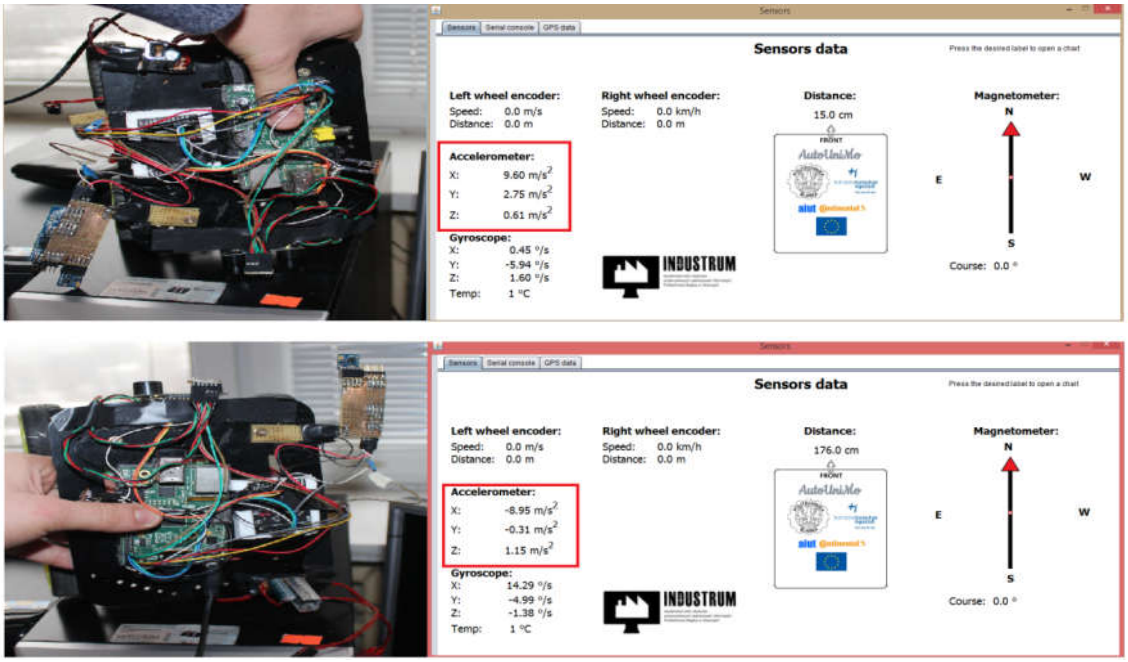

Fig. 9. Test of x-axis
Rys. 9. Test osi x

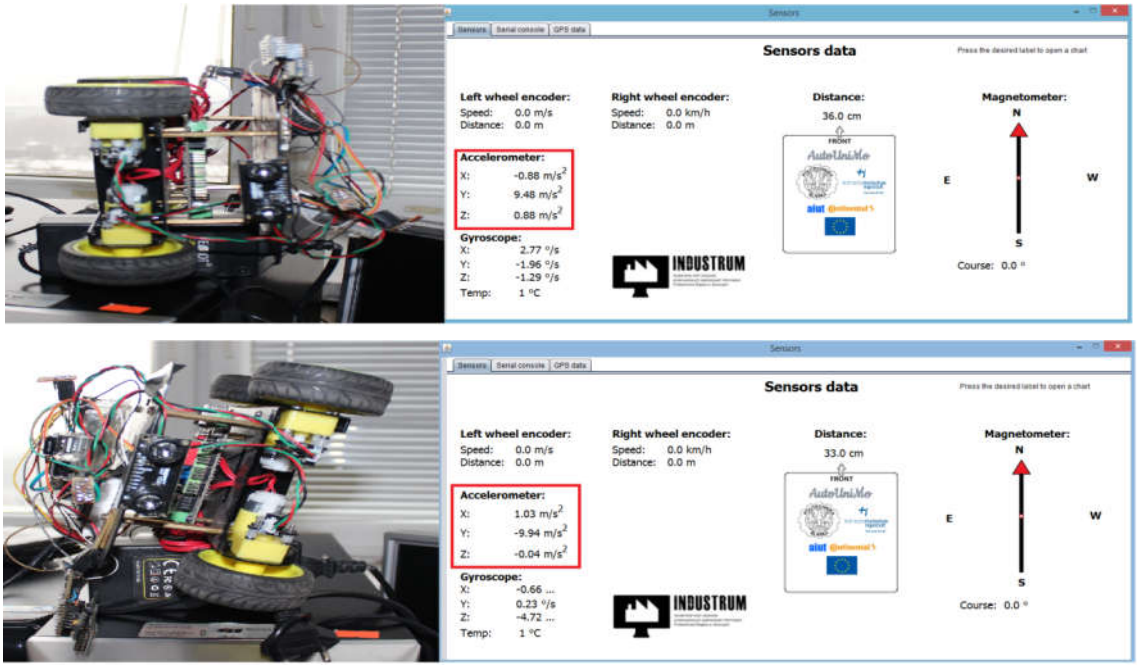

Fig. 10. Test of y-axis
Rys. 10. Test osi y

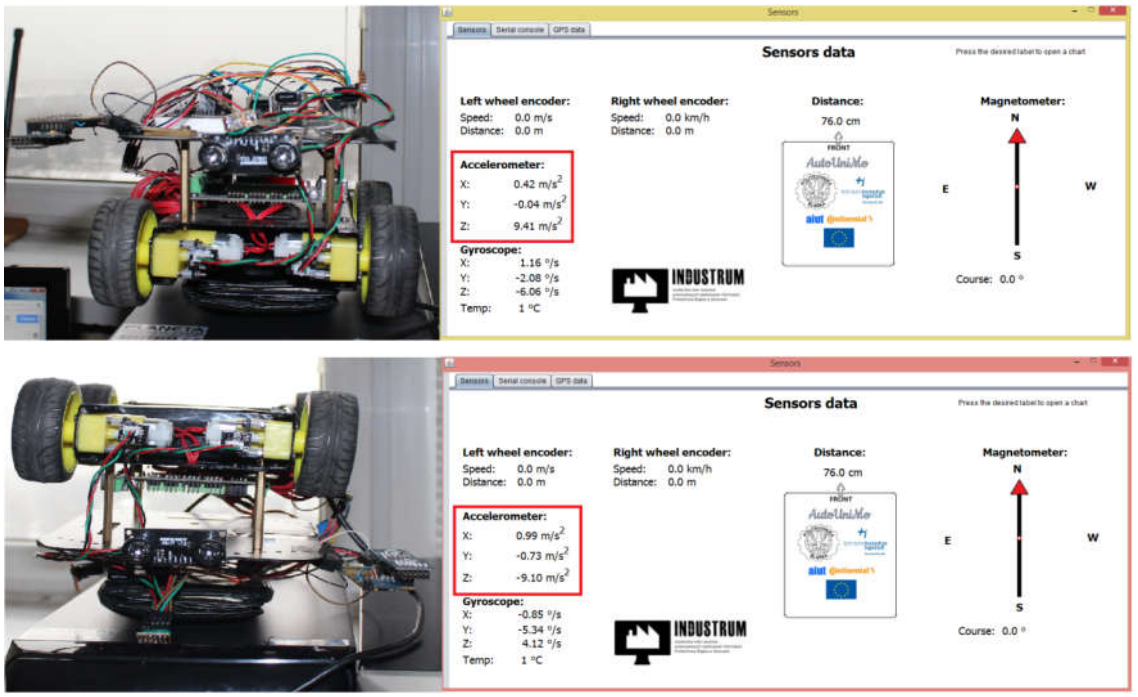

Fig. 11. Test of z-axis
Rys. 11. Test osi z

Initialize function establish connection between the encoder and platform by setting which pin of Raspberry Pi will be responsible for power the sensor and which for reading the signal from sensor.

Encoder signal output changes state to opposite (from HIGH to LOW or vice versa) when the wheel rotates. Transition takes place 20 times per one full rotation.

Program counts peaks number of transitions and basing on it calculates values of distance and speed.

To calculate the distance following formula was used:

$$distance=[(2*3.14*wh\_Radius)/20]*cnt \quad (4)$$

where wh_Radius represents the value of radius of wheel connected with encoder(in meters), cnt (impulses counted by the Rasppberry Pi internal counter) is the field of class created (by the authors) in C++ for encoder programming.

In program instead of (2*3.14)/20 was used value 0.31415 which is the same, but handmade calculated to minimize computational complexity of formula.

First was calculated length of wheel divided by resolution of encoder (20). Next this value is multiplied by number of counted peaks. As result value of distance passed by platform in meters was obtained. To calculate the speed following formula was used:

$$Speed=distance_{temp}/time \quad (5)$$

It is almost the same formula as for distance but $distance_{temp}$ is measured in a specified time interval (0.01 s). If the time interval is constant then is enough to divide the value of distance passed in this time by it to obtain speed.

Off function simply turn off sensor.

In Fig.11 was present tests of encoder work.

Thanks to simplicity of this encoder results of measurements in fact depend only on a given radius of wheel, so if this value is accurate then results also are very good. Only in measurement of speed can be seen small error and delay during fast changing of speed. It is because of time needed to make measurement.

4.2. Power management

4.2.1. Requirements

The work is based on an assumption of existing up to 8 sensors or other devices that work on voltage of 3.3 V, next 8 on 5 V and another 8 on 12 (or 16) V. At first glance it can be seen, that sensors and small devices can be powered through Raspberry Pi (pins: 1, 2, 4 supply 5 V and pin 17 – 3.3 V). Unfortunately 12 V have to be supplied from outer source.

This creates another problem - such voltage, when put on any of Raspberry Pi's pin can severely damage the platform, so galvanic isolation is an essential safety objective.

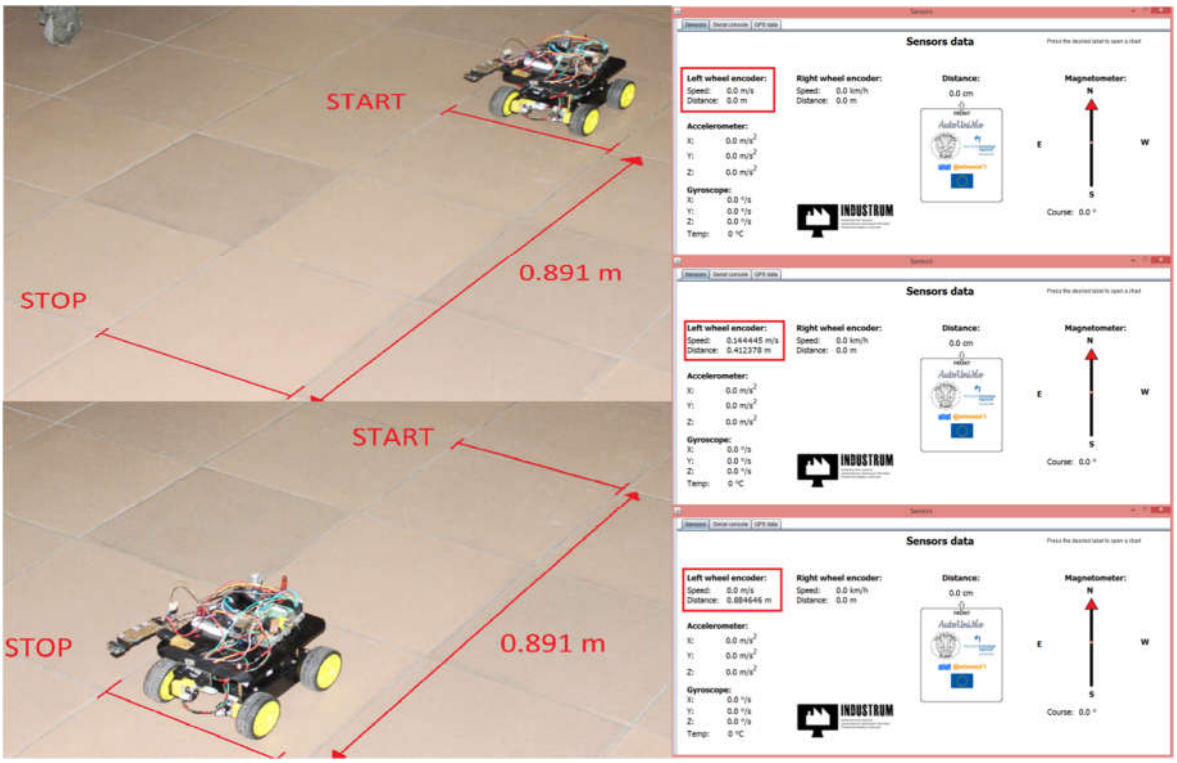

Fig. 12. Test of distance and speed measurements
Rys. 12. Testy pomiaru przemieszczenia i prędkości

Next thing to face is limited number of GPIOs (General Purpose Input/Output). Since there are only 8 available of them - this is too little to control even half of the devices which were meant to be turned on and off. The key requires 24 unique digital pins – "0" for power off, "1" for power on. In order to obtain such number of outputs there can be either another device connected via I2C or shift registers used.

Of course, current cannot be supplied directly from the state defining pins. In order to power a device the state has to be converted into a system which is capable of conducting current. At the end there has to be created a PCB Raspberry HAT on which all the above operations take place.

4.2.2. Application

4.2.2.1. Obtaining of 24 programmable outputs – the key

As described above – to obtain 24 unique output pins the best device which could be used is shift register. It does not require numerous pins to be handled and when connected in series with another register the number of their output pins can reach any value (which is a multiple of 8). The device needs also an output lach, so when the values are set by the register there is no output blinking. The register which fulfills those requirements is 74HC595. All the

operations of seting desired output on all 24 pins require usage of only 3 Raspberry Pi GPIOs.

Shift register's handling C function for the Raspberry Pi:

```
void power(bool *a){
      // CLEARING GPIO STATES
      bcm2835_gpio_clr(LATCH);
      bcm2835_gpio_clr(CLK);

      // SHIFTING DATA
      int i;
      for (i=23; i>=0; i--){
                  switch(key[i]){
                              case 0:
                                          bcm2835_gpio_clr(DATA_IN);
                                          break;
                              case 1:
                                          bcm2835_gpio_set(DATA_IN);
                                          break;
                              }
                  bcm2835_gpio_set(CLK);
                  bcm2835_gpio_clr(CLK);
      }
      bcm2835_gpio_set(LATCH);
      bcm2835_gpio_clr(LATCH);
                              }
```

4.2.2.2. Converting outputs into power switches

Having already provided outputs, which define whether to turn the sensors power on or off there had to be found a method of converting them to some kind of switch. One can come out with the idea of placing sensors on the collector line of the bipolar junction transistor. Such method can be used as long as for conducting current state – saturation mode – voltage drop on U_{CE} is not significant. It has to be remembered, that sensors operate on certain voltage ranges and exceeding those ranges may lead to either wrong readings or even disability to read anything. In example – Pololu ACS711EX for which 3.3 V is typical operating voltage has a minimal supply voltage 3.0 V described in datasheet. HC-SR04 Ultrasonic Sensor for which 5 V is typical, may operate in ranges of 4.5 – 5.5 V. Other devices are even more susceptible to voltage changes.

Another weakness of this application is base current I_B. First of all, the used shift register has a maximum current supply value for its outputs, so we cannot reach I_B higher than some level, what results in I_C also being restricted by a certain value ($I_C = I_B\beta$). Of course, for this projects 300 mA will not be exceeded in case of sensors on a collector lane, so this issue can be omitted. But there arises another problem concerning I_B. For maintaining saturation mode of the transistor (so the sensor is powered) I_B current has to be continually drawn from the batteries, and even if it seems small for one structure it should be remembered that we are supplying current to 24 devices. Now the loss grows to a perceptible level.

One can come up with another idea. Bipolar junction transistors require base current to maintain conduction on collector, but there can be used unipolar ones. They require only

some portion of electrons to load a gate to open I_D and the gate can be grounded when we intend to stop supplying current to sensor. Thanks to it this is not needed to maintain certain value of I_B all the time sensor is running.

The last method of controlling flow of current for a particular sensor was to use relay module. Its strong points are that it converts high/low states we obtain from the shift registers output directly to either open or short circuit for the powering source of the device we want to control. It also provides inbuilt galvanic isolation, so it can be used to control 12/16 V devices without additional precautions. Its big disadvantage is size and delay time. There cannot be used 24 relays, or even only 8 for higher voltages, because the space they occupy is enormous.

In order not to damage Raspberry Pi with voltages exceeding its admissible values, circuits powering certain devices should be isolated from it.

First idea was a previously presented relay It is easy implementable and safe module to use, but as it was written before – too big to be used in the project.

Secondly, there exists tested and commonly used method – optoelectronic isolation. One of the elements which suit project's needs the most is LTV847 – High Density Mounting Type Photocoupler. The element uses optoelectronic elements to separate circuits – current flowing on the input side makes LED shine which light, just like I_B in bipolar transistors, causes output side to conduct current more or less proportional to the one flowing on the diode side. The complete circuit is shown in Fig. 13.

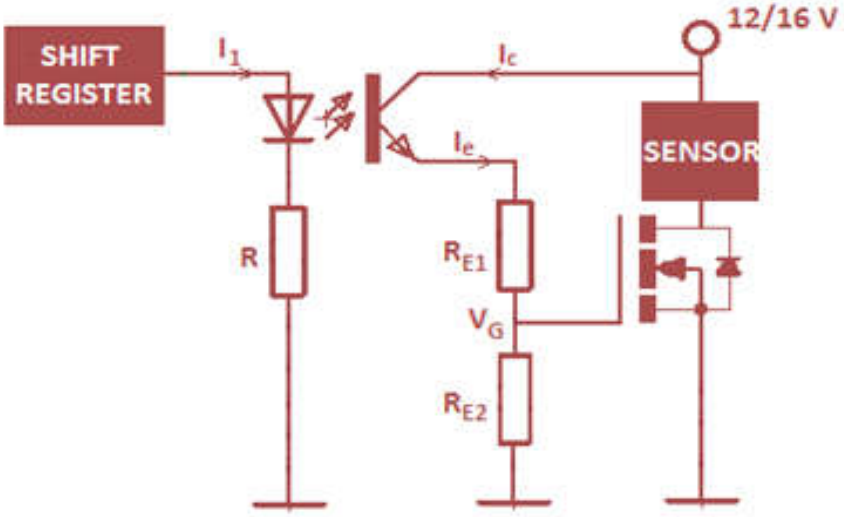

Fig. 13. Galvanic isolation structure
Rys. 13. Schemat układu izolacji galwanicznej

4.2.2.3. Creating Raspberry Pi HAT PCB

When the connections of all components are designed there is only creating Raspberry HAT left. The HAT is a PCB which is designed to be easily installed and removed from

GPIOs whenever usage of the board is required or not. The final layout looks as shown in Fig. 14.

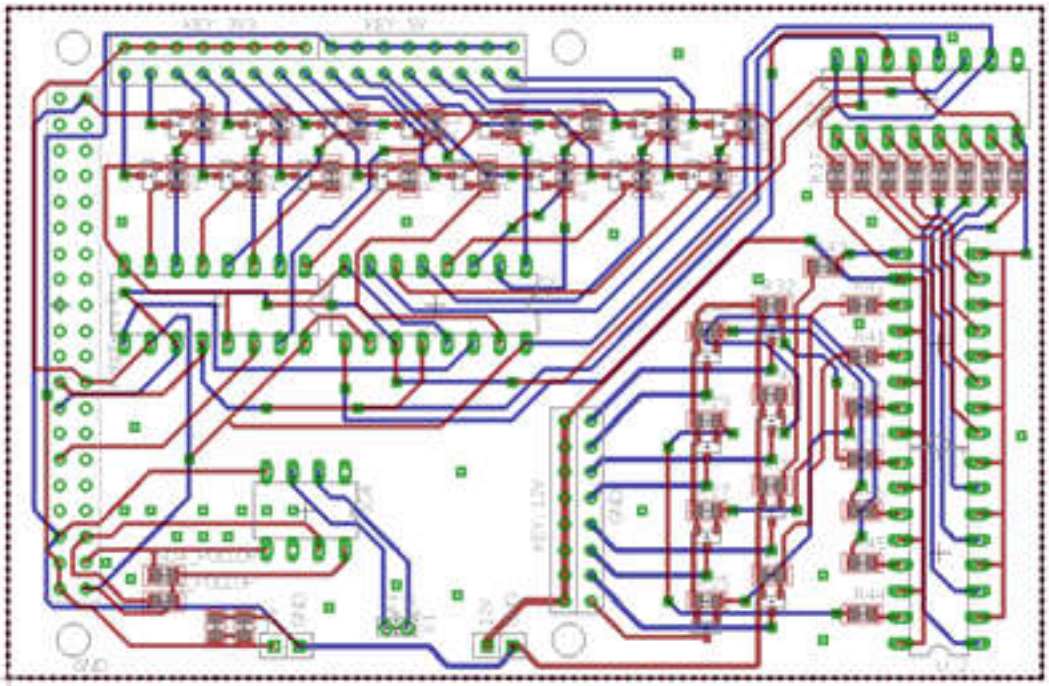

Fig. 14. PCB – board layout
Rys. 14. Widok płytki PCB

5. Conclusion

In this paper, the authors presented a part of AutoUniMo project. Various sensors application is described and the power management for sensors was presented. Sensors and devices, even if not used, draw current from the batteries. So, the idea of reducing the power usage with programmable key was introduced. This would allow the control of current flow for each device that is included in the process. The main platform is the Raspberry Pi Linux based computer and the designed – especially for this platform – HAT (Hardware Attached on Top) does not only provide power supplying key, but it is as well a convenient and neat way to mount devices and sensors. Moreover, future designers of other PCBs can use the CS Eagle library for project development.

Acknowledgements

This work was supported by the European Union from the FP7-PEOPLE-2013-IAPP AutoUniMo project "Automotive Production Engineering Unified Perspective based on Data Mining Methods and Virtual Factory Model" (grant agreement no: 612207) and

research work financed from funds for science in years 2016-2017 allocated to an international co-financed project (grant agreement no: 3491/7.PR/15/2016/2).

BIBLIOGRAPHY

1. Hanke T., Hirsenkorn N., Dehlink B., Rauch A., Rasshofer R., Biebl E.: Generic Architecture for Simulation of ADAS Sensors. Proc. of 16th International Radar Symposium (IRS), 2015, p. 125÷130.
2. Gruyer D., Belaroussi R., Li X., Lusetti B., Revilloud M., Glaser S.: PerSEE: A central sensors fusion electronic control unit for the development of perception-based ADAS. Proc. of 14th IAPR International Conference on Machine Vision Applications (MVA), May 18-22, 2015. Miraikan, Tokyo, Japan, p. 250÷254.
3. Xin J., Zhencheng H., Hsin G.: A new multi-sensor platform for adaptive driving assistance system (ADAS). Proc. of 8th World Congress on Intelligent Control and Automation, June 21-25 2011, Taipei, Taiwan, p. 1224÷1230.
4. Somandepalli V., Marr K.: Thermal Safety Management of Lithium-Ion Battery Energy Storage Systems for Use in Ocean-going and Subsea Applications. OCEANS 2015 – MTS/IEEE Washington, p. 1÷7.
5. Won I.K., Kim D.Y., Hwang J.H., Lee J.H., Won C.Y.: Lifetime Management Method of Lithium-ion battery for Energy Storage System. 18th International Conference on Electrical Machines and Systems (ICEMS), 2015, p. 1375÷1380.
6. Shinde P.A., Mane Y.B.: Advanced Vehicle Monitoring and Tracking System based on Raspberry Pi. IEEE 9th International Conference on Intelligent Systems and Control (ISCO), 2015, p. 1÷6.
7. Mambou E.N., Swart T.G., Ndjiounge A.R., Clarke W.A.: Design and Implementation of a Real-Time Tracking and Telemetry System for a Solar Car. AFRICON, 14-17 Sept. 2015, p. 1÷5.
8. Ziebinski A., Cupek R., Erdogan H., Waechter S.: A Survey of ADAS Technologies for the Future Perspective of Sensor Fusion, in Computational Collective Intelligence: 8th International Conference, ICCCI 2016, Halkidiki, Greece, September 28-30, 2016. Proceedings, Part II, Nguyen T. N., Iliadis L., Manolopoulos Y., Trawiński B. (Eds.). Springer International Publishing, 2016, p. 135÷146.
9. Honeywell HMC5883L Datasheet, February 2013, Form # 900405 Rev E
10. STMicroelectronics L3G4200D Datasheet, December 2010, Doc ID 17116 Rev 3
11. Analog Devices ADXL345, 3-Axis, ±2 g/±4 g/±8 g/±16 g Digital Accelerometer, Data Sheet, Rev. E

Omówienie

W artykule opisano część projektu AutoUniMo traktującą o znaczeniu i wykorzystaniu modułów sensorycznych w mobilnej platformie autonomicznej osadzonej na komputerze Raspberry Pi (Raspbian Linux). W skład projektu wchodzą - akcelerometr, żyroskop, magnetometr, enkoder dla kół oraz Raspberry Pi HAT zaprojektowany m.in. do zarządzania poborem energii.

Platforma zasilana jest z ogniw polimerowo-litowych. Aby zoptymalizować ilość energii potrzebną na obsługę czujników przez moduł Raspberry Pi opracowano klucz elektroniczny, pozwalający na programowe włączanie/wyłączanie zasilania poszczególnych czujników, jak pokazano na Rys. 1. System sensoryczny obejmuje następujące czujniki: magnetometr (Rys. 2) – podłączony z wykorzystaniem interfejsu I^2C, żyroskop (Rys. 3) – do wyboru interfejs I^2C oraz SPI (wybrano I^2C), czujnik przyspieszenia (Rys. 4) – interfejs jak poprzednio, enkoder optyczny (Rys. 5) – impulsy TTL. Działanie wszystkich czujników zostało przetestowane. Stworzono do tego celu dedykowaną aplikację. Przed pomiarami, czujniki – poza enkoderem – poddano prostej kalibracji. Wyniki pomiarów przedstawiono odpowiednio na rysunkach: dla magnetometru, Rys. 6; dla żyroskopu, Rys. 7 i Rys. 8; dla akcelerometru, Rys. 9, 10 i 11; dla enkodera optycznego, Rys. 12. Opracowano prosty przełącznik umożliwiający sterowanie włączaniem i wyłączaniem zasilania poszczególnych czujników (Rys. 13) oraz dedykowaną płytkę drukowaną, umożliwiającą przyłączenie wymaganych układów peryferyjnych (w tym czujników) do modułu Raspberry Pi (Rys. 14).

Addresses

Paweł RYBKA: Silesian University of Technology, Institute of Electronics, ul. Akademicka 16, 44-100 Gliwice, Poland.

Krzysztof WOSIK: Silesian University of Technology, Institute of Electronics, ul. Akademicka 16, 44-100 Gliwice, Poland.

Łukasz SZCZEPAŃSKI: Silesian University of Technology, Institute of Electronics, ul. Akademicka 16, 44-100 Gliwice, Poland.

Adam ZIĘBIŃSKI: Silesian University of Technology, Institute of Informatics, ul. Akademicka 16, 44-100 Gliwice, Poland, adam.ziebinski@polsl.pl.

Rafał CUPEK: Silesian University of Technology, Institute of Informatics, ul. Akademicka 16, 44-100 Gliwice, Poland, rafal.cupek@polsl.pl.

Roman WYŻGOLIK: Silesian University of Technology, Institute of Automatic Control, ul. Akademicka 16, 44-100 Gliwice, Poland, roman.wyzgolik@polsl.pl.

Sebastian BUDZAN: Silesian University of Technology, Institute of Automatic Control, ul. Akademicka 16, 44-100 Gliwice, Poland, sebastian.budzan@polsl.pl.