AKADEMIA GÓRNICZO-HUTNICZA

Wydział Informatyki, Elektroniki i Telekomunikacji

KATEDRA INFORMATYKI

Poduszkowiec

IEiT

Informatyka

Technologie internetu rzeczy

2015/2016

Mateusz Jarosz

Paweł Białas

Daniel Ogiela

Spis treści

1. Temat projektu

Temat projektu to budowa oraz zaprogramowanie sterowania poduszkowca. Należało obsłużyć skręcanie, przyspieszanie oraz wznoszenie/opadanie pojazdu. Projekt jest udoskonaloną wersją wcześniej realizowanego projektu.

2. Harmonogram pracy

Data	Postęp
2016-04-27	Wybór tematu projektu
2016-05-11	Przygotowanie oprogramowania na moduł ESP8266
2016-05-14	Połączenie modułu ESP z silnikami
2016-05-15	Uzyskanie przewodowej kontroli nad sterem
2016-05-18	Stworzenie i uruchomienie aplikacji na system android
2016-05-20	Uzyskanie kontroli bezprzewodowej nad silnikami
2016-05-25	Prezentacja działania urządzenia na zajęciach
2016-06-08	Oddanie projektu

3. Wykorzystywany sprzęt

- ESP8266-07
- 2 x silnik EMAX XA2212
- Śmigło 2 płatowe
- Śmigło 3 płatowe
- 2 x regulator napięcia ESC ABC POWER 30A
- Styrodur 3mm oraz 6mm
- Konwerter USB/Serial

4. Sposób działania poduszkowca

Poduszkowiec posiada dwa silniki. Pierwszy z nich odpowiada za wznoszenie się. Kręcąc śmigłem zagarnia powietrze z nad siebie i kieruje je pod siebie. Tworzy to siłę, a przez to, że powietrze ma

bardzo ograniczone możliwości aby uciec, podnosi poduszkowiec. Dzięki temu silnikowi pojazd ma bardzo znikome tarcie lub kontakt z podłożem w ogóle nie występuje. Drugi silnik odpowiada za napęd. Odpycha powietrze i zgodnie z 3 zasadą z dynamiki sam także otrzymuje pewną siłę (o przeciwnym zwrocie niż ta, którą sam wytworzył).

5. Sposób komunikacji z poduszkowcem

W komunikacji z urządzeniem posłużyliśmy się modułem komunikacji WiFi tj. EPS8266-E07. Tworzymy na nim sieć lokalną o adresie 192.168.4.1 i tworzymy serwer HTTP na porcie 80. Następnie korzystając z urządzenia z systemem Android zaopatrzonym w przygotowaną przez nas aplikację łączymy się z siecią stworzoną przez moduł ESP. Używając aplikacji łączymy się z serwerem przez socket używający protokołu TCP. Następnie cyklicznie co 0.1s wysyłamy do modułu informacje o ustawieniach zadanych przez obsługującego aplikację. Moduł odbiera te informacje, dzieli je i wysyła na odpowiednie lokacje (tj. do steru oraz obu silników).

6. Uruchamianie poduszkowca

1. Podłączyć moduł ESP8266-07 do komputera z zainstalowanym oprogramowaniem Arduino IDE za pomocą konwertera USB/Serial.

2. Wykorzystując Arduino IDE załadować przygotowane oprogramowanie do modułu ESP826607.

3. W przypadku braku błędów powinna się pojawić nowa dostępna sieć.

4. Stworzyć aplikację do sterowania poduszkowcem. Aby to zrobić należy na komputerze z zainstalowanym kompilatorem Python, biblioteką *kivy*, narzędziem *Buildozer* oraz Android SDK wykonać punkt "Programowanie kontrolera do sterowania poduszkowcem".

5. Podłączyć ster na podstawie punktu "podłączanie się do steru i silników". Następnie włączyć aplikację. Jeżeli nie otrzymujemy ekranu pokazanego w punkcie "Programowanie kontrolera do sterowania poduszkowcem", ale aplikacja zostaje natychmiast zamknięta należy sprawdzić czy na pewno jesteśmy podłączeni do odpowiedniej sieci.

6. Podłączyć silniki do modułu ESP8266-07 na podstawie punktu "podłączanie się do steru i silników".

7. Do regulatora należy podłączyć baterię. Jeżeli silnik będzie wydawał ciągłe piknięcia, oznacza to, że nie dostał mu podany stan niski na starcie. Aby to zrobić należy włączyć aplikację i jednym z dwóch suwaków szukać w celu znalezienia odpowiedniego stanu (objawi się to zagraniem przez silnik krótkiej melodii).

W przypadku gdy nie istnieje taka wartość, która daje stan niski to należy przeprogramować regulator (jest to także dobry pomysł nawet po uzyskaniu kontroli nad silnikiem w celu posiadania pełnego zakresu pracy silnika). Aby to uczynić należy podążać zgodnie z poleceniami dostępnymi w instrukcji regulatora (link w bibliografii). W tym momencie uzyskujemy pełną kontrolę nad jednym z silników.

7. Podłączanie się do steru i silników

Aby podłączyć ster należy połączyć następujące kable:

- kabel czarny wychodzący ze steru do pinu GND lub wyprowadzić go na płytkę stykową
- kabel czerwony wychodzący ze steru do pinu 5V na płytce stykowej
- kabel żółty wychodzący ze steru do pinu 12 na płytce ESP

Aby podłączyć silnik podnoszący należy połączyć następujące kable:

- 3 czarne kable wychodzące z tego silnika z 3 kablami wychodzącymi z regulatora w taki sposób, aby kolory zacisków się zgadzały parami (jest to ważne dla prawidłowego działania silnika)
- Kabel czarny wyprowadzony przez wtyczkę żeńską należy podłączyć do pinu GND lub płytki stykowej na wcześniej wyprowadzone miejsce
- Kabel biały wyprowadzony przez wtyczkę żeńską należy podłączyć do pinu 16 na płytce ESP

Aby podłączyć silnik napędowy należy połączyć następujące kable:

- 3 czarne kable wychodzące z tego silnika z 3 kablami wychodzącymi z regulatora w taki sposób, aby kolory zacisków się zgadzały parami (jest to ważne dla prawidłowego działania silnika)
- Kabel czarny wyprowadzony przez wtyczkę żeńską należy podłączyć do pinu GND lub płytki stykowej na wcześniej wyprowadzone miejsce
- Kabel biały wyprowadzony przez wtyczkę żeńską należy podłączyć do pinu 14 na płytce ESP

8. Programowanie modulu ESP8266-07

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>

#include <Servo.h>

//////////////////////
// WiFi Definitions //
//////////////////////
const char WiFiAPPSK[] = "SIECTESTOWA";
const char ssid[] = "HELLOWORLD";
int pos=90;
int vel=30;
int hig=30;
WiFiServer server(80);
Servo myservo,velocity,heigh;
WiFiClient client;

void setup()
{
  myservo.attach(12);
  velocity.attach(14);
  heigh.attach(16);
  WiFi.mode(WIFI_AP);
  WiFi.softAP(ssid, WiFiAPPSK);
  server.begin();
}

void loop()
{

  // Check if a client has connected

  if(!client){
     client = server.available();
  }

  if (!client) {
    return;
  }

  // Read the first line of the request
  String req = client.readStringUntil('\n');

  char  a =  req[req.length()-1];
  pos = int(a);
  char  v =  req[req.length()-2];
  vel = int(v);
  char  h =  req[req.length()-3];
  hig = int(h);
```

```
  if(pos>180||pos<0){
          pos=90;
        }
  myservo.write(180-pos);
  velocity.write(vel);
  heigh.write(hig);

  delay(1);

}
```

Funkcja *setup* wywołuje się tylko raz po podłączeniu modułu ESP8266 do źródła zasilania. Przypisane w niej są sygnały wysyłane do silników i serwa (sterującego sterem) do pinów 12, 14 i 16 oraz następuje inicjacja sieci i ustawienie jej parametrów.

Funkcja *loop* jest wykonywana stale po zakończeniu funkcji setup. Funkcja odbiera żądanie a następnie parsując je przekształca otrzyamne znaki w sposób czytelny dla ich docelowych nadawców (tj. silników lub serwa) a następnie je tam wysyła.

9. Programowanie kontrolera do sterowania poduszkowcem

W przypadku braku posiadania któregoś z narzędzi należy go pobrać. Linki do główny stron można znaleźć w bibliografii.

9.1. main.py

```
from kivy.uix.widget import Widget
from kivy.properties import ObjectProperty
from kivy.clock import Clock
import socket

hval = 0
vval = 0
aval = 90

TCP_IP = '192.168.4.1'
TCP_PORT = 80
BASE = "serwo="

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TCP_IP, TCP_PORT))
```

```
class HeightSlider(Widget):
    pass

class VelocitySlider(Widget):
    pass

class AngleSlider(Widget):
    pass

class HoovercraftDriver(Widget):
    hslider = ObjectProperty(None) vslider =
    ObjectProperty(None) aslider = ObjectProperty(None)

    def update(self, dt):
        global hval, vval, aval
        lhval = hval + 35.0 lvval = vval + 35.0
        s.sendall(BASE + chr(int(lhval)) + chr(int(lvval)) + chr(int(aval)) + "\n")


    def on_touch_move(self, touch):
        if touch.x < self.width / 5:
            if touch.y >= 5 and touch.y <= self.height - 10:
                global hval
                self.hslider.pos = 5, touch.y
                hval = (92.0 * touch.y)/(self.height - 10)

        if touch.x < 2 * self.width / 5 and touch.x > self.width / 5:
            if touch.y >= 5 and touch.y <= self.height - 10:
                global vval
                self.vslider.pos = self.width / 5 + 5, touch.y
                vval = (92.0 * touch.y)/(self.height - 10)

        if touch.x > 2 * self.width / 5:
            if touch.x < self.width - 10:
                global aval
                self.aslider.pos = touch.x, 5
                aval = (180.0 * (touch.x - 2.0/5.0 * self.width))/(3.0/5.0 * self.width - 5.0)


class HeightSlider(Widget):
    pass

class VelocitySlider(Widget):
    pass
```

```
class AngleSlider(Widget):
    pass

class HoovercraftApp(App):
    def build(self):
        app = HoovercraftDriver()
        Clock.schedule_interval(app.update, 1.0 / 10.0)
        return app

if __name__ == '__main__':
    HoovercraftApp().run()
```

9.2. hoovercraft.kv

```
#:kivy 1.0.9

<HeightSlider>:
    canvas:
        Rectangle:
            pos: self.pos
            size: self.size

<VelocitySlider>:
    canvas:
        Rectangle:
            pos: self.pos
            size: self.size

<AngleSlider>:
    canvas:
        Rectangle:
            pos: self.pos
            size: self.size


<HoovercraftDriver>:
    hslider: h_id
    vslider: v_id
    aslider: a_id

    HeightSlider:
        id: h_id
        pos: 5, 5
        size: root.width / 5 - 10, 5
```

```
    VelocitySlider:
        id: v_id
        pos: root.width / 5 + 5, 5
        size: root.width / 5 - 10, 5

    AngleSlider:
        id: a_id
        pos: root.width * 7 / 10 - 3, 5
       size: 6, root.height – 10
```

9.3. buildozer.spec

```
title = HooverCraft Driver
package.name = main
package.domain = org.test source.dir
= .
source.include_exts = py,png,jpg,kv,atlas
version = 1.0
requirements = kivy
orientation = landscape
fullscreen = 1
android.permissions = INTERNET
log_level = 1
warn_on_root = 1
```

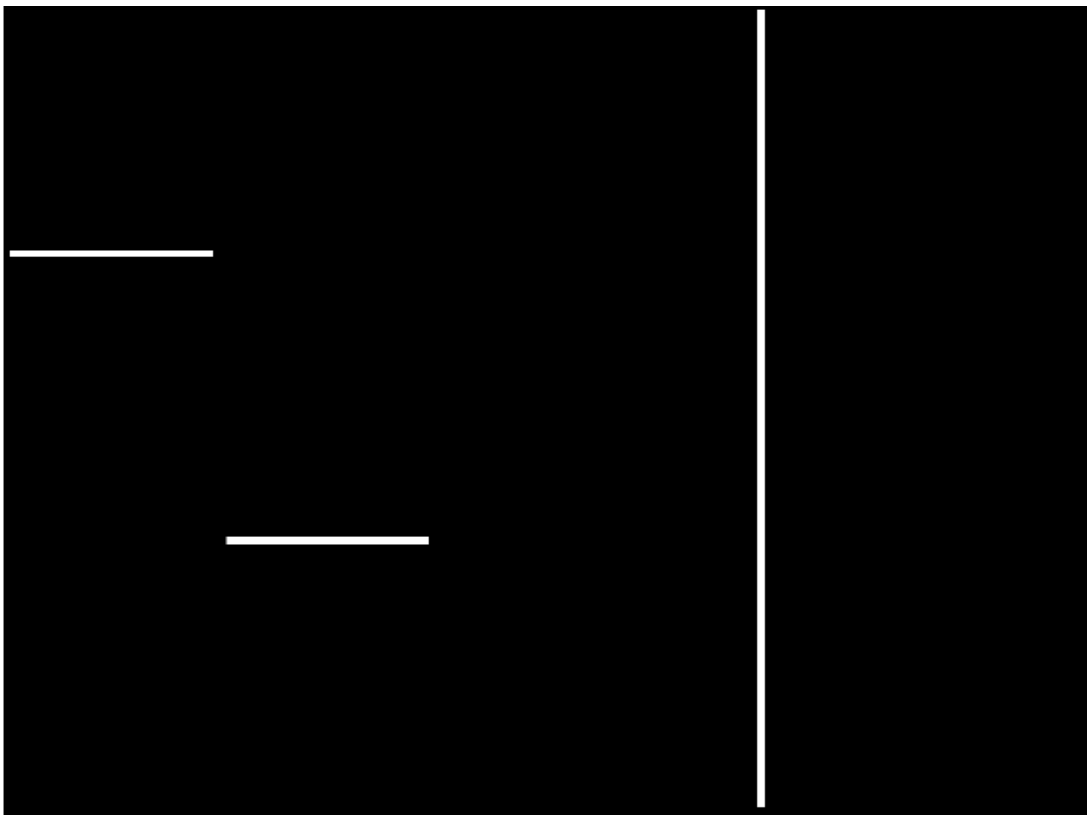
Plik *main.py* zawiera główną część aplikacji. Klasa HoovercraftApp tworzy jeden obiekt typu HoovercraftDriver oraz zajmuje się pilnowaniem wysyłania danych do modułu. Klasa HoovercraftDriver zawiera referencję na obiekty które mają zajmować się obsługą serwa, silnika wznoszącego i silnika napędowego. Metoda *update* jest wywoływana co 0.1s i wysyła ona przez otwarty socket na adres i port naszego serwera na module ESP stringa postaci "serwo=hva\n", gdzie h jest to typ znakowy określający na jakiej wysokości ma znajdować się poduszkowiec (czyli wartość dla silnika wznoszacego), v czyli prędkość, z jaką poduszkowiec ma się poruszać (informacja dla silnika napędowego) oraz a czyli kąt jaki ma być ustawiony na serwo (czyli informacja dla steru). Metoda *onTouch* zajmuje się ustawieniem jednego z trzech suwaków w odpowiednim miejscu i ustawieniu odpowiedniej zmiennej globalnej (z której później informacje bierze funkcja *update*.

Plik *hoovercraft.kv* zawiera informacje o tym jak dana aplikacja powinna wyglądać czyli jeżeli rozpatrzymy tę aplikację z perspektywy wzorca projektowego *Model-View-Controller* to zajmuje się tylko i wyłączenie *view* (a main.py model i controller).

Plik *buildozer.spec* jest generowany przez narzędzie buildozer. Zawiera on konfigurację, która jest informacją dla rzeczonego już buildozera, którego głównym zadaniem jest stworzenie pliku .apk dzięki któremu możemy zainstalować i używać napisana przez nas aplikację na

systemach typu Android (buildozer umożliwia też budowanie aplikacji na urządzenia z systemem Apple, ale jest to dopiero wersja niestabilna i jest obecnie rozwijane.

Wygląd aplikacji:



Pierwszy pasek od lewej ma szerokość jednej piątej szerokości ekranu. Biały prostokąt może być przesuwany po całej wysokości ekranu. Odpowiada on za wysokość podnoszenia się poduszkowca.
Drugi pasek od lewej ma także szerokość jednej piątej szerokości ekranu. Odpowiada on za prędkość poruszania sie poduszkowca.
Ostatnia strefa odpowiada za sterowanie kierunkiem lotu poduszkowca. Jego zakres zaczyna się od dwóch piątych szerokości ekranu (patrząc od lewej strony) aż do prawej krawędzi ekranu. W przeciwieństwie do poprzednich dwóch suwaków, jego położenie może zmieniać się na płaszczyźnie poziomej (a nie pionowej jak w przypadku dwóch poprzednich).

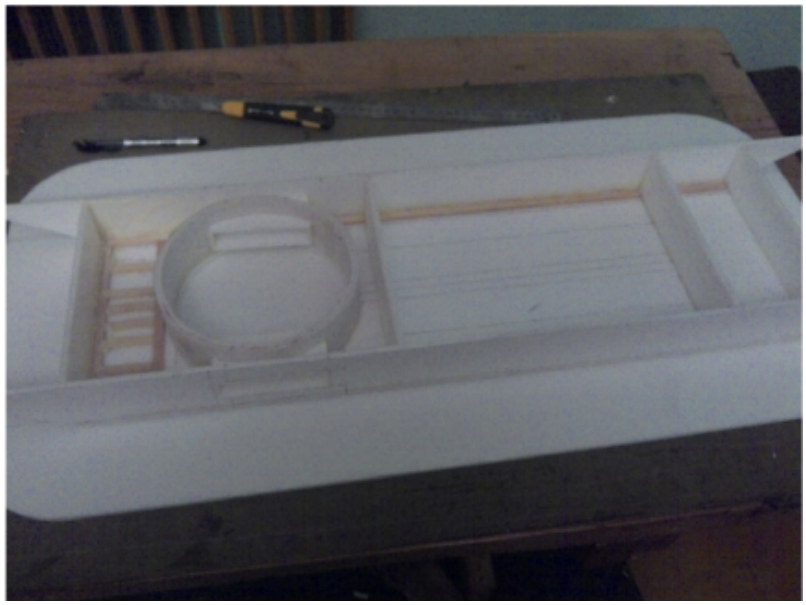
10. Elementy projektu zrealizowane we wcześniejszej wersji wykorzystane w obecnej

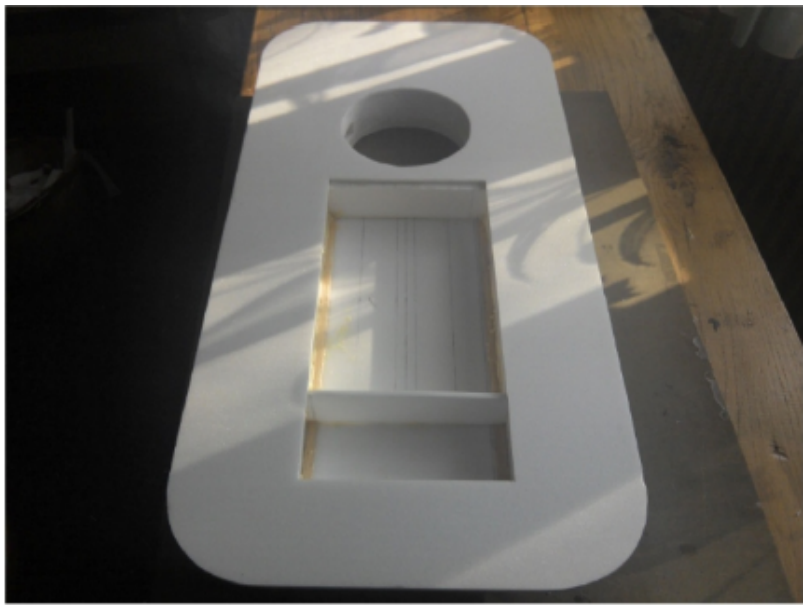
10.1. Składanie głównej części modelu

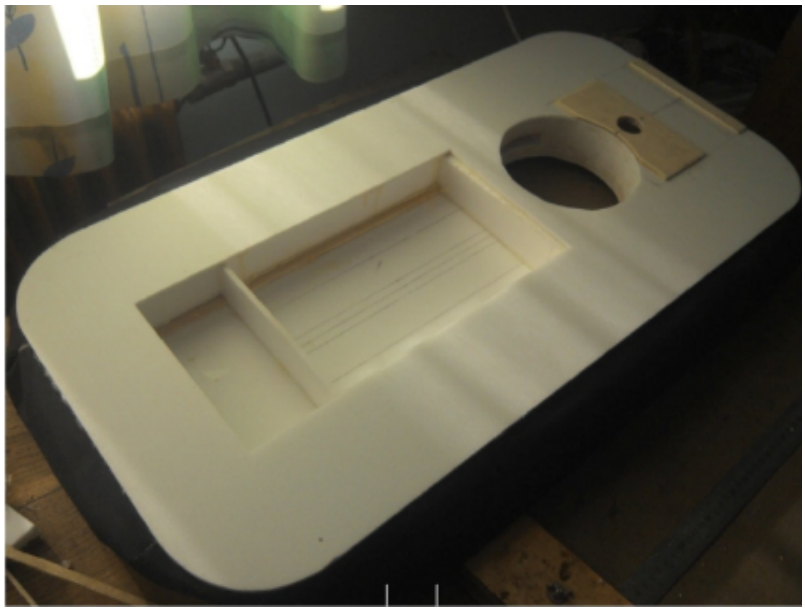
Projekt rozpoczęty został od zbudowania głównej części modelu tj. pojazdu, który po podłączeniu do niego innego sterowania nadawałby się do jazdy.

Pod poniższym linkiem można znaleźć próby sterowania poduszkowcem po podłączeniu do niego zewnętrznego modułu sterowania

http://student.agh.edu.pl/~mateusja/mikro/video/







10.2. Uzyskanie bezprzewodowej kontroli nad sterem

Pierwotnym zamysłem było stworzenie kontrolera przez stronę WWW. Jednakże pomysł ten legł w gruzach ponieważ jQuery, które chcieliśmy zaprząc do przesyłania danych, nie poradził
sobie z taką prędkością przesyłania danych. Jako zamiennik zdecydowaliśmy się zmienić sposób przesyłania danych na proste wysyłanie na socket TCP.

10.3. Uzyskanie kontroli nad sterem w dużej prędkości oraz stworzenie aplikacji na system Android do kontroli steru

Po napisaniu aplikacji na system Android przeprowadziliśmy testy. Można je znaleźć pod następującym niepublicznym linkiem:

https://youtu.be/tRewklb0bVI

10.4. Uzyskanie kontroli bezprzewodowej nad silnikami

Ze względu na to, że ruch, który jest główną dziedziną naszego projektu, ciężko uchwycić na zdjęciach to kręciliśmy filmy. Poniższy film przedstawia próby sterowania poduszkowcem z podłączonymi silnikami

https://youtu.be/StERsE_sHQQ

10.5. Ukończenie budowy owiewki kadłuba

Owiewka kadłuba ma za zadanie osłonić pokrywę miejsca, gdzie jest schowana elektronika oraz chronić silnik podnoszący przez ewentualnymi latającymi przedmiotami bez zmniejszania jego ciągu i dostępu powietrza.

10.6. Malowanie modelu i organizacja kabli

Model pomalowaliśmy na kolory żółty i pomarańczowy ze względu na jego dużą widoczność. Użyliśmy specjalnej farby w spreju przeznaczonej specjalnie do malowania styropianu.

11. Sterowanie

Sterowanie poduszkowcem jest trudnym wyzwaniem ponieważ elementy skrętne są usytuowane z tyłu modelu. Przypomina to ciągłą jazdę autem na biegu wstecznym. Nie pomaga także fakt, że najlepsze efekty uzyskuje się jadąc z pewną prędkością.

Warto tutaj także wspomnieć, że użyte w projekcie silniki są bardzo silne i przeważnie ustawianie pracy obu poniżej połowy jest wystarczające. Ustawianie powyżej 3/4 mocy jest odradzane i użytkownik robi to na własną odpowiedzialność (ponieważ przy takim ciągu śmigła prędkość jaką uzyskuje poduszkowiec jest niszcząca zarówno dla konstrukcji wieży bądź fartucha), nie wspominając o niezwykle trudnym manewrowaniu sterem przy takiej prędkości ponieważ serwo może mieć problemy wykonywaniem ruchów.

12. Wykorzystanie

Urządzenie zostało zaprojektowane pod kątem wykorzystania go przede wszystkim w celach rozrywkowych. Możliwość poruszania się poduszkowca w wielu rodzajach warunkach terenowych pozwala na sterowanie poduszkowcem w różnych okolicznościach. Powyższa oferta skierowana jest do osób pragnących przyjemnego spędzenia wolnego czasu.
Wspomniana możliwość poruszania się poduszkowca w różnorakich warunkach terenowych pozwala na wykorzystanie go w celu dotarcia do trudno dostępnych miejsc – tereny grząskie, bagienne itp. Opisaną właściwość można wykorzystać poprzez umieszczenie na poduszkowcu kamery i wirtualne zwiedzanie trudno dostępnych miejsc.

13. Bibliografia i przydatne linki

1. Instrukcja do regulatora ABC-POWER:
   http://abc-rc.pl/templates/images/files/995/1384185241-instrukcja-esc-abc-power-4881.pdf
2. Dokumentacja Pythona:
   https://www.python.org/doc/
3. Android SDK:
   http://developer.android.com/sdk/index.html
4. Arduino IDE:
   https://www.arduino.cc/en/Main/Software
5. Forum ESP8266:
   http://www.esp8266.com/
6. Kivy:
   https://kivy.org/
7. Buildozer:
   https://github.com/kivy/buildozer