

TDK DOLGOZAT

Készítette:

Müller Péter János



PÉCSI TUDOMÁNYEGYETEM
UNIVERSITY OF PÉCS

Pécsi Tudományegyetem

Pollack Mihály Műszaki és Informatikai Kar

Műszaki Informatika Tanszék

Quadcopter vezérlés HMI rendszer segítségével

Készítette: Müller Péter János

Mérnök Informatika MSc

Konzulens: Dr. Schiffer Ádám,

Egyetemi adjunktus

Pécs

2015

Tartalom

1.	Bevezetés	5
2.	A megoldandó feladat leírása	6
3.	A szemmozgás követés során alkalmazott megoldások.....	6
4.	Az aktuális szabályozási környezet bemutatása	9
5.	A szabályozási folyamat menete.....	10
6.	A szabályozás során alkalmazott képfeldolgozási eljárások	10
7.	Az Androidos alkalmazás bemutatása.....	13
7.1.	A fejlesztőkörnyezet rövid bemutatása	13
7.2.	Az alkalmazás bemutatása.....	14
7.2.1.	A felhasználói felület rövid leírása	14
7.2.2.	Az alkalmazás elkészítéséhez használt megoldások	15
8.	A LabVIEW interface	19
9.	A drón rövid ismertetése	22
10.	A drón szabályozására szolgáló almodul működése	24
10.1.	A drón szabályozásáért felelős almodul bemutatása.....	25
11.	Az elkészült rendszerek összekapcsolása	29
11.1.	Manuális szabályozás.....	29
11.2.	A képfeldolgozásból nyert adatok.....	32
11.3.	A szenzorokból nyert adatok	33
12.	A főprogram bemutatása	34
13.	Elkészült rendszer eredményeinek bemutatása.....	35
14.	Jövőbeli tervek, elképzelések	37
15.	Összegzés, konklúzió	37
16.	Ábrajegyzék	38

17.	Hivatkozások	39
18.	Források.....	40

1. Bevezetés

A dróntechnológia kezdetben csak a filmekben és katonai akciókban alkalmazott eszköz volt. A repülésre alkalmas legtöbb esetben kis gépeket katonai, taktikai célpontok megfigyelésére alkalmazták. Segítségükkel képesek voltak titkos, valamint az emberek számára elérhetetlen helyek megközelítésére. A technológia fejlődése során több típusuk is elterjedt. A fizikai jellemzők, valamint alkalmazhatósági területek segítenek ezek osztályozásában. Beszélhetünk merevszárnyú UAV egységekről, melyeket rendszerint nagy távolságok megtételére, megfigyelési feladatok elvégzésére és légi felvételek elkészítésére alkalmaznak. A másik nagy csoportot a rotorokkal felszerelt drónok teszik ki, melyek neve a telepített rotorok számától függ. Ez alapján beszélhetünk quad-, hexa-, és octokopterekről. Az ilyen egységek rövid távolságok, magasságok megtételére alkalmas kisebb eszközök. Előnyük, hogy a merevszárnyú egységekkel ellentétben képesek arra, hogy egy helyben is lebegjenek. A szabályozásról egy összetett távirányító gondoskodik, melynek kezelését egy kiképzett, tapasztalt személy látja el. A szabályozáshoz továbbá egy külön kijelzőre is szükség van, ahol a vezérlő személy figyelemmel kísérheti a jármű haladást, valamint a látottak alapján beavatkozhat, utasítást adhat.

A műszaki és intelligens eszközökre jellemző, hogy – a fejlődéssel párhuzamosan – a korábban csak kutatásra és hadászatra alkalmazott eszközök a mindennapi életünkbe is begyűrűznek. Ez a drón technológiában sem történt másként. Az elterjedés hátterében nemcsak a széleskörű alkalmazhatóság (épületfelmérés, katasztrófa sújtotta területek felmérése, életmentés, játék) áll, hanem a szabályozást megkönnyítő fejlesztések is.¹² A korábban már említett több rotoros gépek szabályozása, repülésdinamikai okok miatt, korábban nem volt megoldható, de a modern, gyors szabályozó algoritmusok erre is megoldást nyújtottak. A nagy vezérlő egységek mérete csökkent, és megjelentek a kijelzővel felszerelt távirányítók is, valamint egyes eszközök már a sokak által ismert és alkalmazott okos mobiltelefonokkal és táblagépekkel is vezérelhetők.

¹ <http://bigthink.com/design-for-good/the-ambulance-drone-that-could-increase-the-chance-of-survival-after-a-heart-attack-from-8-to-80> (letöltve: 2014.11.19.)

² <http://www.amazon.com/b?node=8037720011> (letöltve: 2014.11.19.)

2. A megoldandó feladat leírása

A dolgozatomban vezérelni kívánt eszköz, a Parrot ArDrone 2.0. drón. A drón, másik nevén quadcopter egy 4 rotorral felszerelt, a kereskedelmi forgalomban bárki számára elérhető eszköz. Mindemellett olyan architektúrával rendelkezik, amely lehetővé teszi, hogy a már ismertetett technológián kívül, alternatív módon is vezérelhessük. Ehhez a megoldáshoz egy korábbi projektben megvalósított technológiát és eljárást alkalmaztam: a szem mozgásának követését valós idejű feldolgozással. A korábbi mérések igazolták, hogy az alkalmazás kellően pontos adatokkal szolgál a szem mozgását illetően, és ez által alkalmas komplexebb környezetben való felhasználásra is.³

Ezzel a fentiekből kirajzolódik dolgozatom témája, amely egy drón szabályozását teszi lehetővé az emberi szem mozgásának monitorozásával, ezzel megalkotva a címben is említett HMI (Human Machine Interface) rendszert, magyarul ember-gép közötti kapcsolatot. A dolgozat többek között tárgyalja a korábban alkalmazott mérési környezetet és eljárást, az alkalmazott képfeldolgozási algoritmusok leírását, a drón és a szabályozó alkalmazás közötti kommunikációt, valamint az elkészült alkalmazást.

A szemmozgás követés segítségével a szabályozási feladat csak részben megoldott, ezért a teljes funkcionalitáshoz egy Androidos eszköz szenzor adatai kerülnek felhasználásra. Ehhez szükséges volt egy Androidos alkalmazásra, valamint egy szabályozó oldali interface megalkotására.

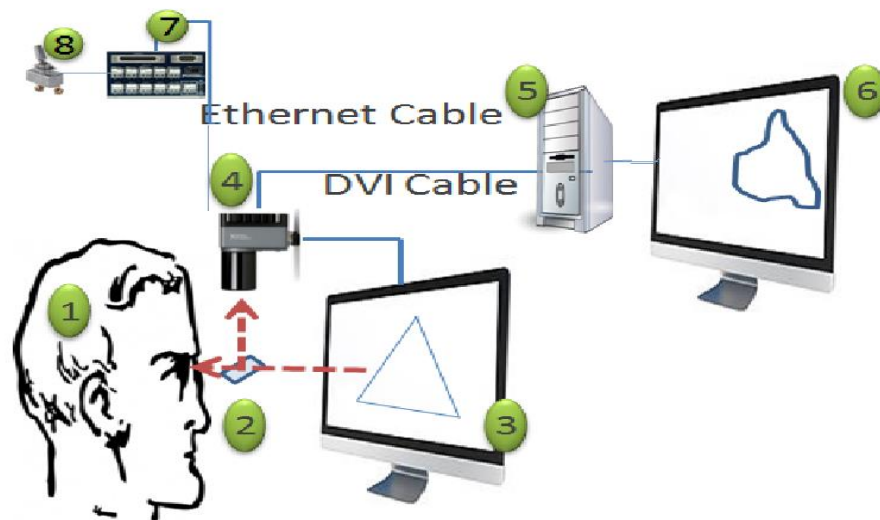
Az alkalmazás elkészítéséhez a National Instruments LabView 2013-as szoftverét, valamint a *Vision* modult alkalmaztam. Az Androidos alkalmazás megalkotásához az Android Studio 1.0.2-es verzióját használtam fel.

3. A szemmozgás követés során alkalmazott megoldások

A következő fejezetben a korábban említett szemmozgás követésére alkalmas rendszer kerül bemutatásra. A megoldandó feladat az volt, hogy egy olyan rendszer kerüljön megalkotásra, ahol egy alany bevonásával képesek vagyunk tekintetének monitorozására

³ Müller Péter: Szemmozgás követő rendszer fejlesztése (szakdolgozat) 2012.

egy adott kijelzőn, és indikálni tudjuk azt az idő függvényében. A rendszer megalkotásához egy külön a feladathoz tervezett mérési kör került kialakításra. Ez magában foglalja a mérési pontosságot elősegítő álltámaszt, valamint a mérési eljárást is. A korábbi mérési folyamat környezete az 1. ábrán tekinthető meg.



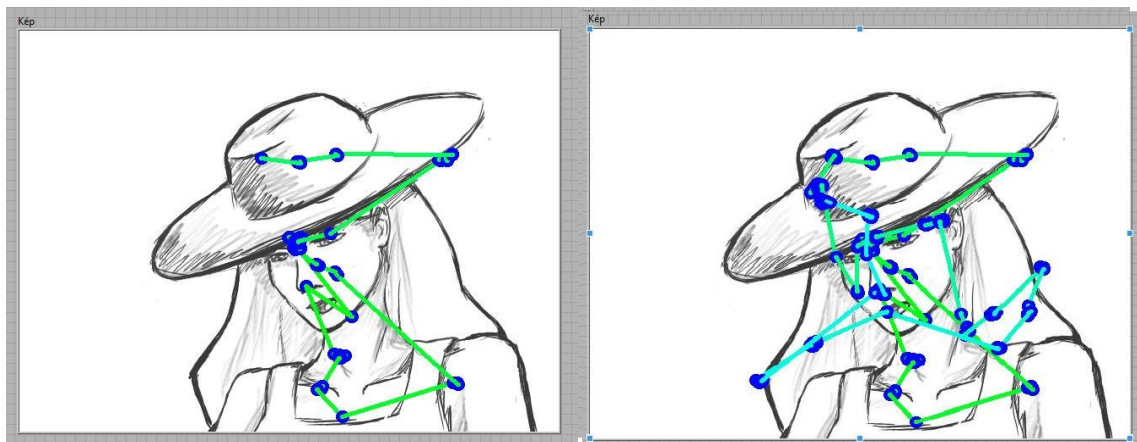
1. ábra: A szemmozgás követéséhez alkalmazott mérési kör

Az ábrán jól látható a tesztalany (1), aki egy félig áteresztő lencsén (2) a másodlagos kijelzőn (3) megjelenő ábrákat adott utasítás szerint megtekinti, ezt a szemmozgást egy erre alkalmas intelligens kamera (4) rögzíti, és továbbítja egy feldolgozó számítógép (5) felé, amely segítségével a szem útja követhető, és indikálható a vizsgálatot végző személy számára (6). Az intelligens kamera nyújtotta lehetőséget kihasználva gombeseménnyel vezérelt felvételezést végeztünk (7,8). A mérési eredmények azonnal megjeleníthetők a vizsgáló személy számára, valamint a későbbiekben a log állományokból visszaállíthatók.

A mérés körben ismertett eszközök segítségével a mérés az alábbi módon zajlik. Az alany elhelyezkedése után ismertetjük a mérés menetét. A mérési folyamatot megelőzően az erre a célra megalkotott kalibrációs pontok felvétele történik. Ennek segítségével a monitoron látott kalibrációs pontokhoz valós koordinátákat rendelünk egy 2D koordináta interpolációs műveletet alkalmazva. A kalibrációs pontok megtekintése után már regisztrált méréseket végzünk, először még csak egyszerűbb geometriai alakzatokon. Ezek segítségével újra ellenőrizhetjük, hogy megfelelőek-e a felvett pontok, valamint itt lehetőség van a mérési pontosság, más néven a hiba, számszerűsítésére. Ezt követően az

alany több ábrát is lát, amelyet tetszése szerint járhat be a tekintetével. A mérés végezetével rendelkezésre áll az információ, amely megmutatja, hogy adott képeken milyen pontokat és régiókat tekintett meg a vizsgált személy.

A mérési eredményeken jól látható, hogy a pontok kirajzolása az idő függvényében történik, amelynek múlását színátmenettel indikáljuk. Az 2. ábrán látható a pontokkal történő reprezentáció, ezt követően a 3. ábrán látható a szemmozgás és tekintetvizsgálatnál használt „heatmap” reprezentáció.⁴⁵



2. ábra: A szemmozgás mozgásának útjának indikálása az idő függvényében



3. ábra: Heatmap reprezentáció

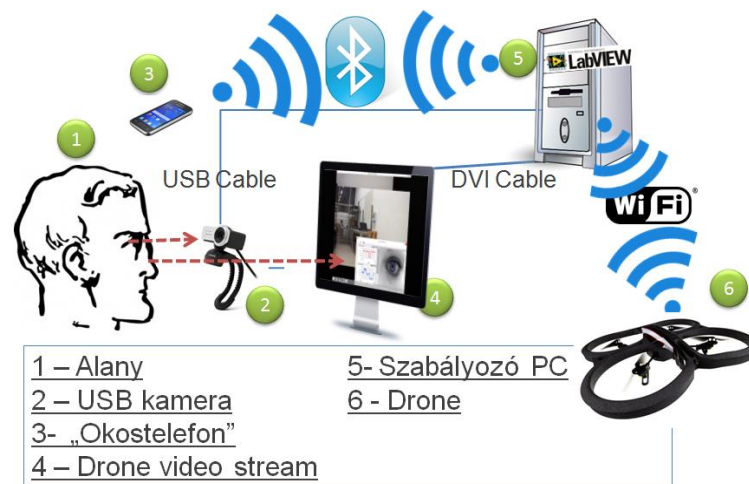
Ebben az esetben is színekkel indikáljuk az érdeklődési pontokat, de itt nem pontokkal, hanem színintenzitással alkotott foltokat látunk a képen. A piros a kisebb intenzitást, a sárga, illetve fehér pedig a legtöbbször megtekintett régiókat mutatja meg.

⁴ Müller Péter: NI pályázat 2013.

⁵ <http://www.zdnet.com/article/eye-tracking-web-usability/>

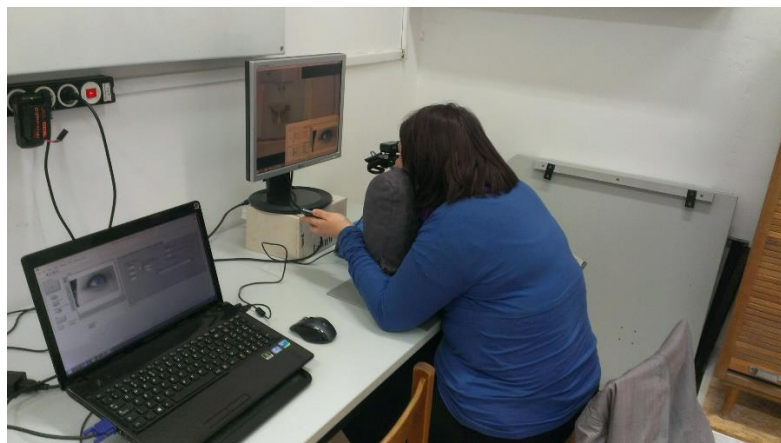
4. Az aktuális szabályozási környezet bemutatása

A mérési környezet megalkotásakor figyelembe kellett venni a korábbi mérés során felmerülő problémákat és azokat a problémákat, amik a mérési folyamat során gondot okozhatnak. A választott szabályozási feladathoz elengedhetetlen a korábbi pontosság, valamint stabilitás megtartása, de szükség van arra is, hogy kellően gyors és könnyen kalibrálható alkalmazást készítsünk el. A 4. ábrán látható az új mérési kör, melynek elemei részben megegyeznek a már korábban is megbízhatóan alkalmazott, a szemmozgás követéséhez használt, mérési környezettel.



4. ábra: A drón irányításához alkalmazott mérési környezet

A mérési kör főszereplője ismét az alany (1), akinek tekintetét egy HD felbontású kamerával (2) rögzítjük, valamint a kézben tartott okostelefon segítségével monitorozzuk, és bluetooth kapcsolaton továbbítjuk a szenzoradatokat (3). A tekintet ezúttal nem egy adott ábrára, illetve nem pontokra, hanem a drón videó adatfolyamára (4) irányul, amit egy másodlagos kijelzőn tekinthető meg. A kamerától kapott videó adatfolyam a szabályozásért felelős számítógépre (5) kerül, ahol a később ismertetett képfeldolgozási eljárásokat követően, egy a szabályozáshoz készült almodul segítségével, a drón (6) számára érthető utasításokat küld szabványos vezeték nélküli hálózati kapcsolat segítségével. A mérőkör megvalósítását az 5. ábra mutatja.



5. ábra: A mérőkörnyezet megvalósítása

5. A szabályozási folyamat menete

A mérés megkezdése előtt ismételten a szabályozásra készülő személy elhelyezkedése, valamint a szükséges eszközök előkészítése az első lépés. Ide sorolható az álltámasz, valamint a szabályozáshoz alkalmazott Androidos eszközön futó alkalmazás előkészítése. A kalibráció ebben az esetben nem kalibrációs tábla segítségével történik, hanem egy referencia középpont regisztrálásával. Ennek segítségével gyorsan és könnyen elvégezhető kalibrációs műveletet hajthatunk végre, ami megkönnyíti a szabályozási feladatot, mégis kellő pontosságot nyújt az eljáráshoz. A referencia középpont felvételét egy gomblenyomással rögzítjük. Ezt követően az élőképből elkészítünk egy mintaképet a vezérlő személy pupillájáról, amit a későbbiekben felhasználunk a képfeldolgozás során. Az előkészületek elvégzése után a felszállást követően a vizsgált személy a monitor két szélső pontját megtekintve képes a drónt balra-jobbra fordítani, a magasságot pedig a képernyő felső, illetve alsó pontjaira nézve szabályozhatja. A kézben tartott eszköz döntésével a drón elmozdulása adható meg. A repülés lezárásához a drónnak kiadjuk a leszállás parancsot, majd a landolást követően a vezérlési folyamatot befejezettnek tekintjük.

6. A szabályozás során alkalmazott képfeldolgozási eljárások

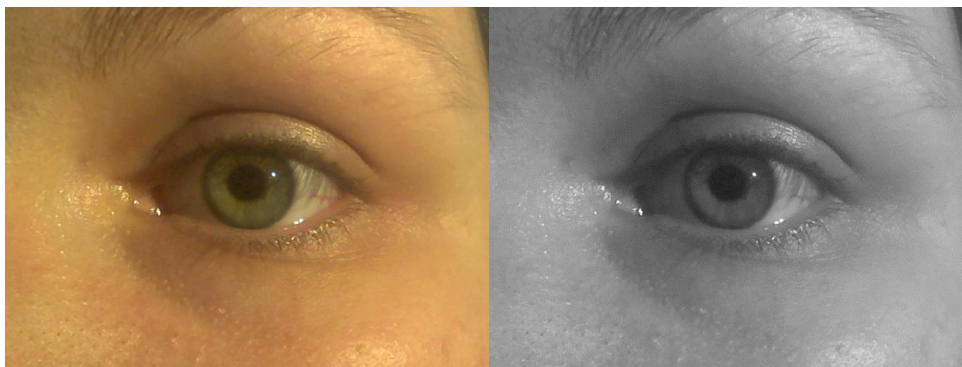
A vezérléshez használt mérési folyamatokhoz a szemmozgás követés során is alkalmazott lépések sorozatára volt szükség. Ebben a fejezetben kerül ismertetésre azon lépések

sorozata, amik lehetővé teszik, hogy a közönséges HD webkamera, mely a 6. ábrán látható, képéből pontos koordináta értékeket kapjunk a szem középpontját illetően.



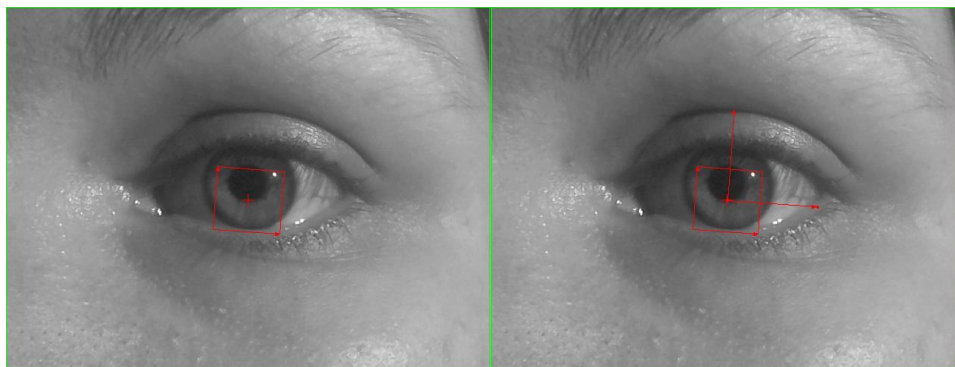
6. ábra: A felhasznált webkamera képe

Első lépésként a programban inicializáljuk a kamerát, majd ezt követően a kamera képén alapvető képfeldolgozási eljárásokat végzünk a pontosabb feldolgozás érdekében. A 7. ábrán látható, hogy az RGB színtartomány HSV tartományba konvertálása, ezzel a mérés szempontjából szükséges, kiértékelhető szürkeárnyaltos képet alkot. A szürkeárnyaltos képtartomány alkalmazása növeli az alkalmazott eljárások pontosságát, mert csökkenti a „false positive” találatok számát.



7. ábra: RGB és HSV színtartományokban látható szem

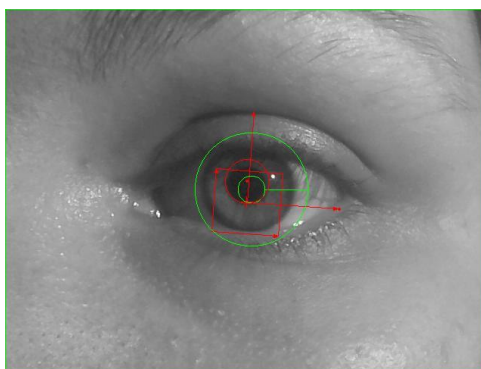
Ezt követően a mintaillesztési eljárást alkalmazva keressük meg a videó adatfolyamon a pupilla régióját. A szabályozási eljárás során rögzített pupilla minta képét felhasználva meghatározásra kerül egy ROI (Region of Interest), amelyet egy adott koordinátájú négyszög fog jelölni, ez látható a 8. ábrán. Ezen a négyszögön belül kell majd keresnünk a pupilla középpontját. A mintakereséshez használt eljárás több paraméterezhető lehetőséget is nyújt. Ezek közé tartozik a minimális egyezési érték, maximális találati szám, valamint a skála és rotáció invariancia.



8. ábra: A pupilla megtalálása a képen

A mintaillesztést követően egy körkeresési eljárást alkalmazunk, amelyet a ROI középpontján definiált koordináta rendszerre illesztünk, így rendelkezésünkre állnak a pupillaközéppont koordinátái. A kör alakzat keresése során ismételten több paraméter érhető el. Ide sorolhatók a körök tulajdonságai, többek között a színátmenetek közti érzékenység, tónus különbségek, illetve a keresési módszerek is a találat eléréséhez (belülről-kifelé, kívülről-befelé). Az objektum köré írt kör tulajdonságai szükség esetén szintén módosíthatóak. A 9. ábrán megfigyelhető a szemközéppont megtalálásának képe.⁶

7



9. ábra: A szem középpontjának megtalálása

A képfeldolgozásért felelős almodul a fent leírt lépéseken kívül még egy fontos feladatot lát el. A kiolvasott koordináta értékeket, amelyek a szabályozáshoz elengedhetetlenek a főprogram szintjére juttatja.

⁶ <http://www.ni.com/pdf/manuals/371007b.pdf> (letöltve: 2014.11.20.)

⁷ <http://www.ni.com/pdf/manuals/371007a.pdf> (letöltve: 2014.11.20.)

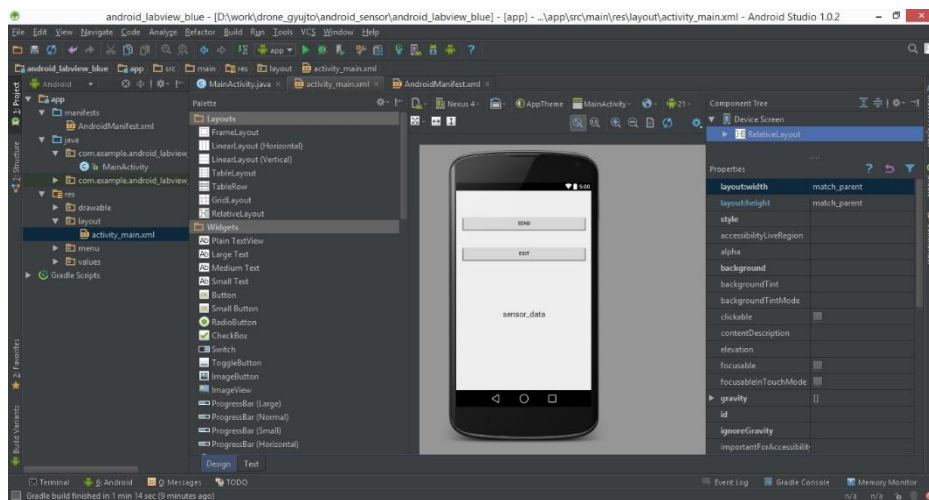
7. Az Androidos alkalmazás bemutatása

A 6. fejezetben bemutatott képfeldolgozási folyamat segítségével, mint már korábban említettem, csak 2 dimenziót vagyunk képesek elérni. Ennek segítségével a drón csak egy adott pozícióban vezérelhető. A megmaradó mozgási formák eléréséhez, szükség van újabb eszköz bevonására, amely képes adatokkal szolgálni a kívánt irányvektorokat és elmozdulási formákat illetően. Ehhez ideálisak az olyan szenzorok, amelyek valós időben képesek megmutatni egy adott eszköz orientációját a térben. Számos erre a feladatra elérhető szenzorból választhatunk, de a cél az volt, hogy költségkímélő, illetve könnyen beszerezhető eszközt válasszak. A választás egy napjainkban igen elterjedt okos, "Smart", eszközre esett, mert már az olcsóbb kategóriájú termékek is rendelkeznek ilyen szenzorokkal. Az okos eszközök két fő csoportja, az Android, illetve az iOS termékei közül az előbbi választottam. A szabályozás során felhasznált eszköz egy SAMSUNG Galaxy Young 2 típusú készülék volt.

7.1. A fejlesztőkörnyezet rövid bemutatása

A fejlesztés során az Android Studio 1.0.2-es verzióját alkalmaztam, amelynek felhasználói felülete a 10. ábrán szerepel. A választást indokolja, hogy egy meglehetősen stabil és átlátható környezetet biztosító fejlesztőkörnyezetről van szó. A fejlesztőkörnyezet alapja a jól ismert ECLIPSE IDE, továbbá a korábban hozzá külön feltelepíthető plugin, az Android SDK környezet. Az IDE-ben megtalálhatóak az alkalmazás megírásához szükséges modulok, grafikus tervező felület, valamint lehetőség van emulátoron történő tesztelésre. Alkalmazás fejlesztés során kiválasztható az API level, amely magában foglalja a futtatáshoz használt eszköz szoftververzióját (JellyBean, KitKat), valamint az eszközökön elérhető függvények aktuális listáját. A fejlesztő környezet folyamatosan figyeli az aktuális kódot és figyelmeztet a kód hibáira, illetve lehetőség szerint javítja azokat. Ezek a funkciók is gyorsítják a fejlesztési folyamatot, valamint gyorsabb hibakeresést tesznek lehetővé.⁸

⁸ <http://developer.android.com/sdk/index.html>



10. ábra: A fejlesztőkörnyezet képe

7.2. Az alkalmazás bemutatása

7.2.1. A felhasználói felület rövid leírása

Az alkalmazás tervezése során az egyszerűséget és a funkcionalitást vettem figyelembe, így a grafikus felületen csak 3 elem kapott helyet, kettő gomb és egy TextView.

A SEND gomb a folyamatos szenzoradat közlésére szolgál. A gomb funkcióját illetően egészen addig továbbít adatokat a szabályozó gép felé, amíg a gomb lenyomása tart. A felengedés pillanatában az adatfolyam megáll.

Az EXIT gomb pedig az alkalmazás bezárásáért felelős. Ilyenkor a megnyitott kapcsolat lezárásra kerül, az alkalmazás tevékenysége leáll.

Az említett TextView az aktuális X és Y értékének valós idejű változása látható. Az alkalmazás előnézete a 11. ábrán található.



11. ábra: Az alkalmazás előnézete Android studioban

7.2.2. Az alkalmazás elkészítéséhez használt megoldások

A kitűzött cél az alkalmazás elkészítése során az volt, hogy lehetőséget nyújtson a szenzoradatok elérésére, valamint a bluetooth adatkapcsolat felépítésére és alkalmazására az adatok továbbításához. Az alkalmazás megalkotása előtt figyelembe kellett vennem az adott API levelhez tartozó szabványok és aktuálisan élő függvények meglétét, valamint azt is, hogy a megfelelő példaprogramok segítségével ki tudjak küszöbölni minden előforduló hibaforrást.^{9 10}. Ahhoz, hogy az alkalmazás ezeket a feladatokat ellássa, engedélyezni kell a bluetooth kapcsolathoz szükséges funkciót. Ezeket a privilégiumokat egy XML állományban állíthatjuk be, az úgynevezett AndroidManifest állományban. A beállításához az alábbi kód szükséges.

```
1. <uses-permission android:name="android.permission.BLUETOOTH" />  
2. <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

Az alkalmazás használatához szükség van a korábban már bemutatott felhasználó felületre. A grafikus felülethez használhatjuk a tervező nézetet, ahol a kívánt elemeket Drag&Drop módszerrel felhelyezzük a Formra, illetve kód formájában is elkészíthetjük. Mindkét esetben egy XML állományt kapunk, ami tartalmazza a felületen található

⁹ <http://developer.android.com/reference/packages.html>

¹⁰ <http://digitalhacksblog.blogspot.de/2012/05/android-example-bluetooth-simple-spp.html>

elemeket azonosítóját és paramétereit. Az alábbi kódrészlet a SEND gomb leírására szolgál.

```
1.      <Button
2.          android:layout_width="wrap_content"
3.          android:layout_height="wrap_content"
4.          android:text="SEND"
5.          android:id="@+id/send"
6.          android:layout_alignParentTop="true"
7.          android:layout_alignParentLeft="true"
8.          android:layout_alignParentStart="true"
9.          android:layout_marginTop="53dp"
10.         android:layout_alignParentRight="true"
11.         android:layout_alignParentEnd="true" />
```

A grafikus felület megtervezését követően az alkalmazás megalkotása következik. Az Android alapú alkalmazásoknál is alkalmazhatjuk az OOP programozási metodikát és annak sajátosságait. Az alkalmazáshoz, amely az Android programok esetében egy java állományt jelent, több függvénykönyvtár importálása is szükséges, de az adott elemek meghívását követően a fejlesztőkörnyezet ebben is a fejlesztő segítségét szolgálja. Deklarálnunk kell a Bluetooth kapcsolathoz szükséges BluetoothAdaptert, BluetoothSocketet, amelyek az inicializálás és kapcsolathoz szükséges a küldés során használt OutputStream-et. A szenzor adatok eléréshez is megadjuk az elengedhetetlen változókat, amelyek magukban foglalják a szenzor típusát, ami jelen esetben egy gyorsulásmérő (accelerometer) és egy SensorManager.

A bluetooth kapcsolathoz, illetve az adattovábbításhoz az úgy nevezett Serial Port Profile (SPP) kliens szerver modellt választottam. Ebben az esetben a kapcsolathoz meg kell határozni egy egyéni azonosítót, amelynek neve Universally Unique Identifier (UUID). Ez egy 128 bites egyéni azonosításra szolgáló, a program szintjén stringként tárolt generált kód. A kód generálására az interneten is több oldal is elérhető. Ez az egyéni azonosító kellően hosszú ahhoz, hogy egy folyamatot azonosítson, valamint szinte lehetetlen, hogy ütközés következzen be. A kommunikációhoz az alkalmazás a Radio Frequency Communication (RFCOMM) protokollt használja, ami egyszerű, de megbízható adatfolyam közlést biztosít a felhasználónak hasonlóan a TCP protokollhoz.¹¹

¹¹ <http://developer.android.com/guide/topics/connectivity/bluetooth.html>

A szabályozó számítógépben található bluetooth kapcsolatot ellátó eszköz, a vezeték nélküli hálózati adapterhez hasonlóan, rendelkezik egy Media Access Control (MAC) címmel, amely egy hexadecimális számsorozat. Ezzel az azonosítóval az eszköz egyértelműen azonosítható. Az alkalmazásban jelenleg ez egy „beégetett” funkció, de egy Plain Text változó, amely egy szövegbeviteli mező segítségével módosítható.

A kapcsolatért felelős adattagokon kívül a grafikus felületen helyt kapó elemekhez is változókat rendelünk, melyek segítségével a kívánt műveletek elvégezhetjük. A megalkotott változók segítségével a gombokhoz eseményfigyelőket rendelve függvényeket hívhatunk meg azok állapotától függően.

Az alkalmazás a futása során több előre definiált állapotba kerül. Az úgy nevezett Activity, magyarul tevékenység, a megalkotás első fázisában választható. A kiválasztott tevékenységhez a fejlesztő környezet alapértelmezett függvényeket rendel. Többek között ilyen például az onCreate függvény. Ennek célja, hogy az elindulás után a benne található utasításokat végrehajtsa. A bluetooth kommunikációra készített alkalmazásban itt történik a fent említett adattagok példányosítása, értékadása (adapter, szenzor, gombok), valamint vizsgálja azt is, hogy elérhető, illetve aktív-e a szolgáltatás.

```
1.      @Override
2.      public void onCreate(Bundle savedInstanceState) {
3.
4.          super.onCreate(savedInstanceState);
5.          setContentView(R.layout.activity_main);
6.
7.          send = (Button) findViewById(R.id.send);
8.          stop = (Button) findViewById(R.id.stop);
9.
10.         btAdapter = BluetoothAdapter.getDefaultAdapter();
11.         checkBTState();
12.         sm = (SensorManager) getSystemService(SENSOR_SERVICE);
13.         accelerometer = sm.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
14.         sm.registerListener(this, accelerometer, SensorManager.SENSOR_D
ELAY_NORMAL);
15.         sensor_data = (TextView) findViewById(R.id.sensor_data);
16.     }
```

A program másik nagyon fontos függvénye a gyorsulásmérő szenzor állapotváltozását figyel, melynek neve onSensorChanged. Ennek segítségével az X és Y irányú elmozdulás vektorok változásakor az aktuális érték egy stringbe kerül. A leírt függvényművelet az alábbi kódrészlet mutatja be.

```

1.      @Override
2.      public void onSensorChanged(SensorEvent event) {
3.          sensor_data.setText("X:" + event.values[0] + "Y:" + event.values[1])
4.          ;
5.          sensor_datastring=sensor_data.getText().toString();
6.      }

```

A kapcsolat során az alkalmazás futása megszakadhat, ezért az alkalmazásban az ilyen esetekben egy figyelmeztető üzenetet küld, úgynevezett Toast üzenetek segítségével. A Toast message egy a képernyőn felugró szövegbuborék, amelyben üzeneteket közölhetünk a felhasználóval.

```

1.      private void errorExit(String title, String message){
2.          Toast.makeText(getApplicationContext(), title + " - " + message, Toast
3.          .LENGTH_LONG).show();
4.          finish();
5.      }

```

A küldéshez a SEND gombot alkalmazzuk. A SEND gomb lenyomását követően az előállított stringet a definiált bluetooth kapcsolaton keresztül küldjük tovább a servernek, amely egy LabVIEW-ban implementált alkalmazás. A küldés folyamatos mindaddig, amíg az alany a gombot lenyomva tartja. A felengedés pillanatában az adatfolyam küldése megáll. A küldés során alkalmazotthoz hasonlóan itt is üzenetek segítségével tájékoztatjuk a felhasználót. A gomb mechanizmusát az alábbi kódrészlet reprezentálja.¹²

```

1. send.setOnTouchListener(new View.OnTouchListener() {
2.     private Handler mHandler;
3.     @Override public boolean onTouch(View v, MotionEvent event) {
4.         switch(event.getAction() & MotionEvent.ACTION_MASK) {
5.             case MotionEvent.ACTION_DOWN:
6.                 if (mHandler != null) return true;
7.                 mHandler = new Handler();
8.                 mHandler.postDelayed(mAction, 100);
9.                 Toast.makeText(getApplicationContext(), "Start send", Toast.L
10.                ENGTH_SHORT).show();
11.                 break;
12.
13.             case MotionEvent.ACTION_UP:
14.                 if (mHandler == null) return true;
15.                 mHandler.removeCallbacks(mAction);
16.                 mHandler = null;
17.                 Toast.makeText(getApplicationContext(), "Stop data", Toast.LE
18.                NGTH_SHORT).show();
19.                 break;

```

¹² <http://ruuhkis.com/android-repeat-action-button/> (letöltve: 2014.12.06.)

```

19.         }
20.         return false;
21.     }
22.
23.     Runnable mAction = new Runnable() {
24.         @Override public void run() {
25.             sendData(sensor_datastring);
26.             mHandler.postDelayed(this, 100);
27.         }
28.     };
29. });

```

A küldés során a már említett Output stream-et alkalmazzuk az adatfolyam továbbításához. A stringként fogadott X és Y értékek egy byte tömbbe kerülnek, és így kerülnek kiírásra. Az adatfolyam küldése az alábbi kódrészleten látható.

```

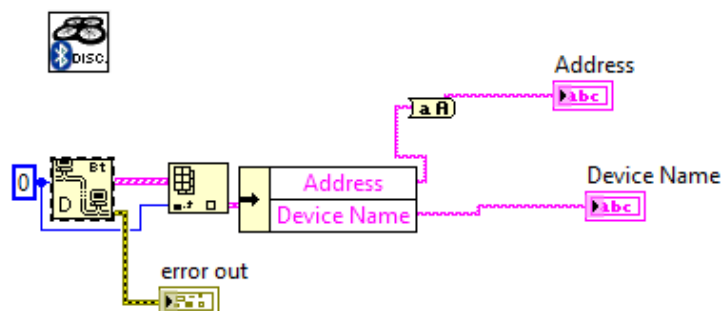
1.     private void sendData(String message) {
2.         byte[] msgBuffer = message.getBytes();
3.
4.         Log.d(TAG, "...Send data: " + message + "...");
5.
6.         try {
7.             outputStream.write(msgBuffer);
8.         } catch (IOException e) {
9.             String msg = "In onResume() and an exception occurred during
write: " + e.getMessage();
10.            if (address.equals("00:00:00:00:00:00"))
11.                msg = msg + ".\n\nUpdate your server address from 00:00:
00:00:00:00 to the correct address on line 35 in the java code";
12.            msg = msg + ".\n\nCheck that the SPP UUID: " + MY_UUID.toSt
ring() + " exists on server.\n\n";
13.
14.            errorExit("Fatal Error", msg);
15.        }
16.    }

```

8. A LabVIEW interface

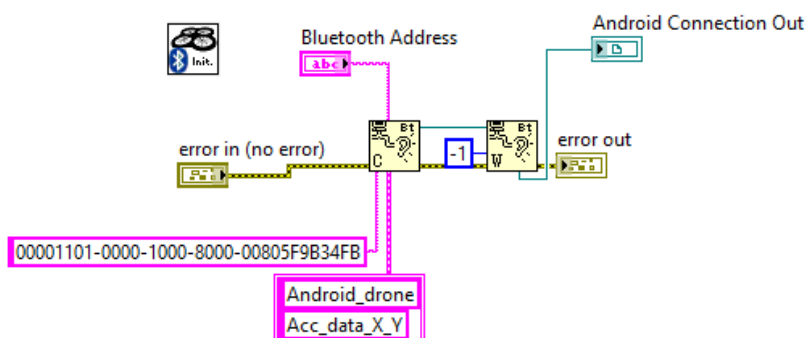
Az előző fejezetben bemutatott alkalmazáshoz szükség van egy fogadó félre. Ezt a feladatot látja el a LabVIEW-ban elkészült interface, amely 5 subvi-ból áll. Ezek feladata a következő: felderítés, inicializálás, adatfogadás, adatkiolvasás, lezárás.

A 12. ábrán látszik, hogy a Discovery vi feladata, hogy kilistázza az elérhető eszközöket. Erre a feladatra a Bluetooth Discovery függvény meghívása alkalmas, amely felderíti és kiírja az elérhető eszközök nevét és MAC azonosítóját.



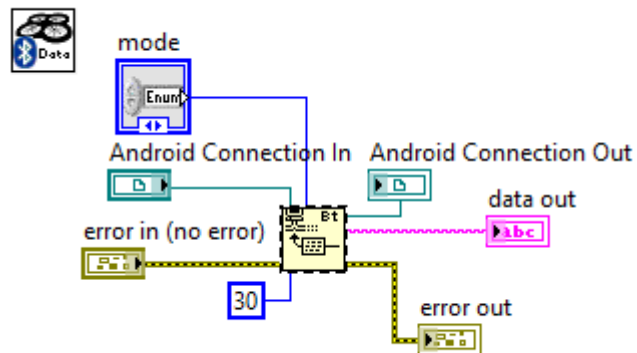
12. ábra: A discovery vi blokk diagramja

Az Init vi feladata – amelynek Block diagramja a 13. ábrán látható – hogy a kiválasztott című eszközt figyelje, és kapcsolatot létesítsen az eszközzel. Ebben a vi-ban Create Bluetooth Listener segítségével az eszköz MAC címéből, valamint a korábban említett UUID stringből megalkotásra kerül egy ListenerID, amit egy Bluetooth Wait On Listener vár. Ennek kimenetén megjelenik a kapcsolatot leíró adattag.



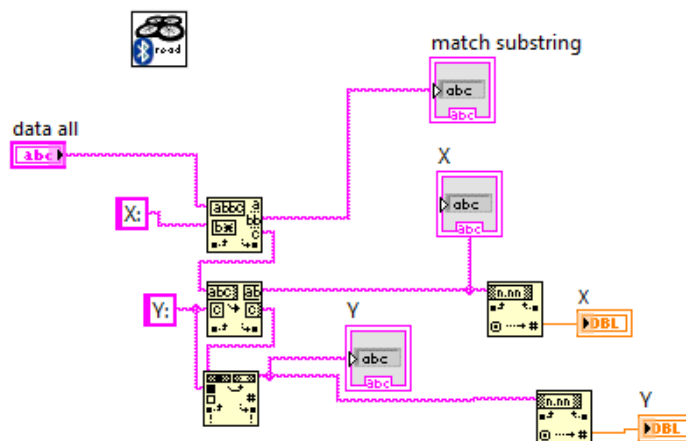
13. ábra: Az init vi blokk diagramja

A Data vi feladata, hogy a kapcsolati adattagban definiáltak segítségével kiolvassa a beérkező adatstringet. Az itt használatos függvény a Bluetooth Read, melynek paraméterként megadható, hogy milyen méretű bitsorozatot olvasson be, illetve az is, hogy milyen módon. Az elkészült subvi a 14. ábrán látható.



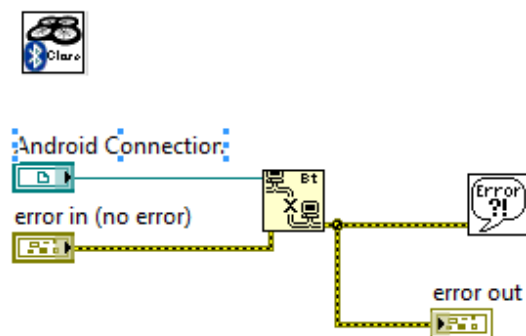
14. ábra: A data vi blokk diagramja

A Read vi feladata, hogy a beérkező stringet felbontsa, így megkapjuk a kívánt szenzoradatokat. Ezt a 15. ábrán látható módon, különböző stringműveletek alkalmazásával érhetjük el.



15. ábra: A read vi blokk diagramja

A Close vi feladata (16. ábra), hogy a kapcsolatot megfelelően lezárja, ezzel kivédve az esetleges hibaforrásokat a következő csatlakozás alkalmával.



16. ábra: A close vi blokk diagramja

9. A drón rövid ismertetése

Az általam vezérelni kívánt eszköz pontos neve AR.Drone 2.0, amit a Parrot francia cég gyárt, és kínál eladásra. A külsejét és fizikai tulajdonságait tekintve könnyű és robosztus kialakítású, karbonszálakkal megerősített vázzal rendelkezik, súlya nem haladja meg a 400 g-ot. Négy darab elektromos léptetőmotor működteti, az áramforrása pedig egy 1500 mAh-os akkumulátor, amely körülbelül 15-20 perc repülést tesz lehetővé.

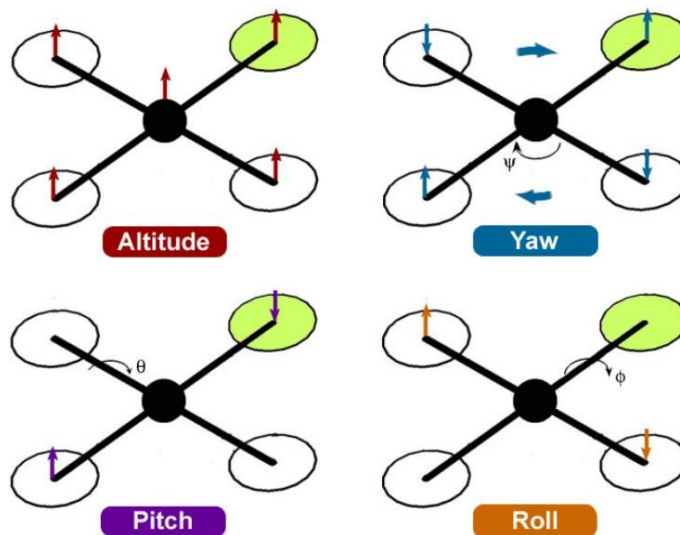
A kopter agyát egy darab 1 GHz-es órajelű 32 bit-es Cortex A8-as processzor biztosítja, amely mellett 1 Gb DDR2-es memóriával rendelkezik. Több kamerával, egy 720p, illetve egy QVGA felbontásúval rendelkezik, továbbá rengeteg szenzorral is bír. Ezek közé sorolható a gyorsulás- és nyomásmérő, ultrahangszenzor, és magnetométer. Ezen erőforrásokat egy Linux rendszer működteti. A felhasznált drón a 17. ábrán látható.¹³

¹³ <http://ardrone2.parrot.com> (letöltve: 2014.11.20.)



17. ábra: A Parrot AR.Drone 2.0 képe

Az, hogy egy ehhez hasonló quadcopter repülni tudjon a szabályástechnika terén történt hatalmas ugrásszerű fejlődésnek köszönhető. Másfél évtizeddel korábban még nem volt lehetőség arra, hogy egy ehhez hasonló egység a levegőbe emelkedhessen, és emberi irányítás alatt képes legyen ott maradni. Ennek az oka, hogy a drón összesen 6 darab szabadsági fokkal, angolul *Degrees of Freedom (DoF)*, rendelkezik, de mi ebből csak 4 mozgási formát vagyunk képesek egyidejűleg irányítani, és elérni. A 18. ábrán látható a négy aktuátor, más néven beavatkozási pont, segítségével elérhető mozgási forma.



18. ábra: A drón irányításhoz szükséges aktuátorok

Az *altitude* paraméter segítségével a magasság változtatható, a *yaw* a saját tengely körüli elfordulást reprezentálja, a *pitch* segítségével a drón előre-hátra mozdul el, végezetül a *roll* segítségével bal, illetve jobb oldal irányú mozgásra képes. A drón haladásához ezek együttes vezérlése szükséges. A magasság változáshoz minden rotor

azonos sebességgel forog, így képes a magasság növelésére és csökkentésére. Az előre-hátra haladás során, a választott iránytól függően, a rotorok ok fordulatszáma páronként nő, illetve csökken. A drón mindig az alacsonyabb fordulatszámú rotor pár irányába halad. A roll, vagyis oldal irányú elmozdulás során, amelynek működési elve megegyezik pitch-csel, csak a bal és jobb oldali rotor párok sebessége változik. Az elfordulást a drón a két kereszt irányú rotor egyidejű fordulatszám növelésével éri el. Itt megemlíthetünk egy 5. mozgás típust, amely a hovering. Ennek feladat, hogy amíg nem érkezik a drón számára olyan parancs, ami helyzetváltoztatásra utasítja, lebegjen a levegőben és tartsa a pozícióját.

10. A drón szabályozására szolgáló almodul működése

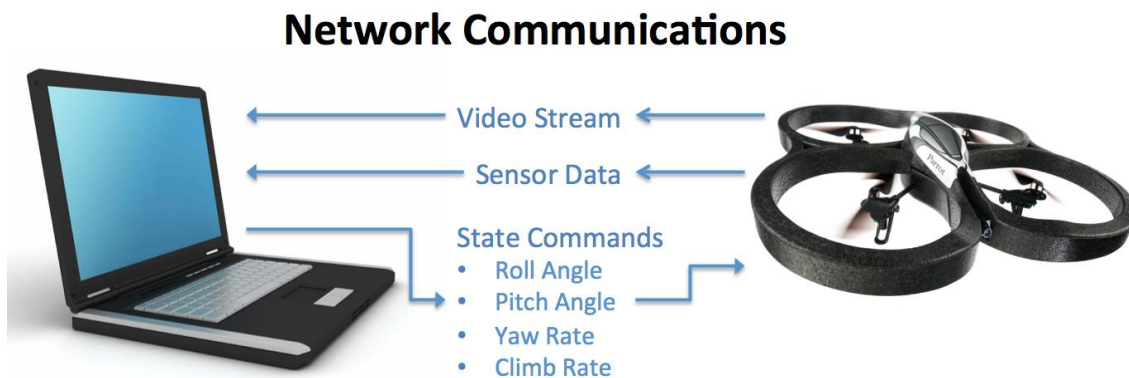
Az eddig leírtakban a korábban alkalmazott mérési, képfeldolgozási lépések feladat specifikussá tétele, valamint az Androidos eszköz szenzoradatainak kinyerése történt meg. Ezen a ponton már képesek vagyunk a vezérlő személy pupillaközéppontjának megtalálására, és az elmozdulási vektorok kinyerésére. A következő fejezete a drón szabályozásához szükséges tudnivalókat, műveleteket tárgyalja.

A drón szabályozásához szükséges feladatok ellátásához szükségünk van annak architektúrájához és szoftveréhez illő környezet kialakítására. Ehhez egy, az interneten elérhető SDK-t és segédletet is alkalmaztam, amely csak az alapvető funkciókat, és szabályozáshoz szükséges paramétereket tartalmazza. Ennek használatát az indokolta, hogy az elkészült részeknél nincs szükség újbóli implementálásra, mert azok a drón saját SDK-jában is dokumentált alapokon nyugszanak, így ez például szolgál a programozásban lehetőség szerint megkívánt újrafelhasználhatóságnak.¹⁴¹⁵ A szabályozásért felelős almodul 3 fő részegységre bontható. Első a drón inicializálásáért felelős, a második a képfeldolgozási, illetve az Android interface modulban említett koordináta értékek fogadására, valamint a szabályozási parancsok küldésére szolgál, a harmadik pedig a drón szabvány szerinti lekapcsolódásáért felelős egységet tartalmazza.

¹⁴ <https://projects.ardrone.org/projects/show/ardrone-api> (letöltve: 2014.11.20.)

¹⁵ https://www.labviewhacker.com/doku.php?id=libraries:lvh_ardrone:tutorials (letöltve: 2014.11.28.)

A következő bekezdés a fázisok részletes bemutatását tárgyalja. Ahogy már korábban említettem a drón és a feldolgozó számítógép között egy előre definiált hálózati kapcsolat segítségével történik a kommunikáció.

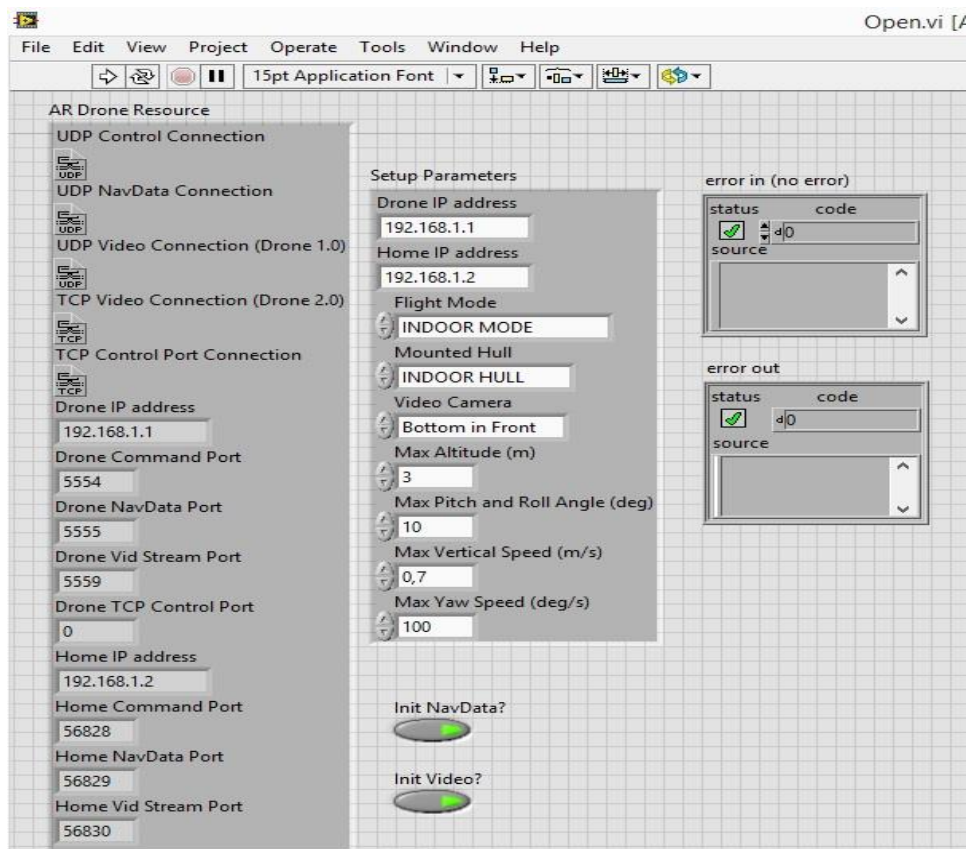


19. ábra: A drón és a számítógép között zajló adatok típusai

A kapcsolat kialakítása nagyon hasonlít egy kliens routerhez történő csatlakozásához. A drón ebben az esetben a router, amelyet az ismert módon egy SSID azonosít, és a vezérlő számítógép pedig a kliens, ami erre csatlakozni képes. Érdeemes megemlíteni, hogy a drón egyszerre csupán egy eszközt képes „kiszolgálni”. A 19. ábrán jól látható, hogy a definiált kapcsolat segítségével milyen adatfolyamokhoz férhetünk hozzá. A programban történő implementációja a következő fejezetben olvasható.

10.1. A drón szabályozásáért felelős almodul bemutatása

Első lépésként megadjuk a drón kommunikációhoz és repülési paramétereket. Ezek az elengedhetetlen erőforrások, amelyeket egy cluster típusba rendezünk. Az inicializálás során be kell állítanunk a drón és a kliens IP címét, valamint a hozzájuk tartozó küldő és fogadó portokat. Ilyen portok például a Video Stream, NavData, valamint a vezérlő parancsok. A következő ábrán látható az inicializálásért felelős vi Front Panel részlete. Az alkalmazás e részének blokk diagramján a megadott paraméterekkel UDP, valamint TCP kapcsolatok nyitása történik, amelyekre az UDP Open, és a TCP Open Connection funkciókat alkalmaztam. A kapcsolati függvényeket a környezet által leírt feltételek mellett kerültek megalkotásra.

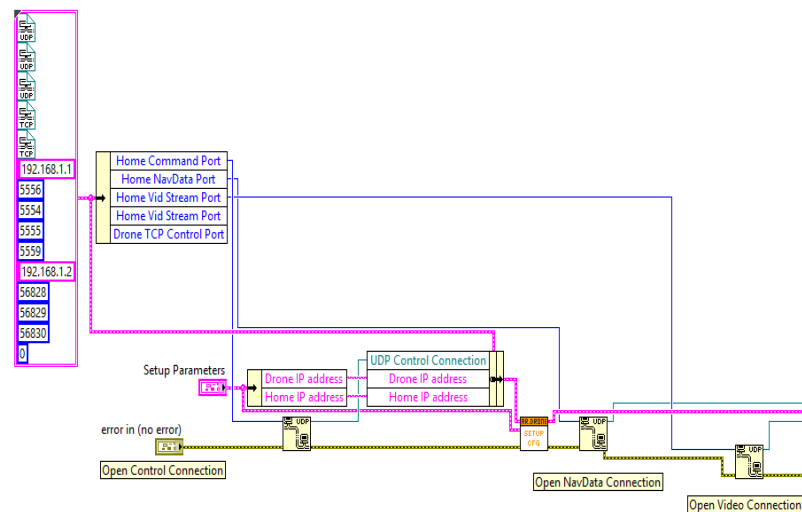


20. ábra: A drón inicializálás paramétere

A 20. ábrán megfigyelhető, hogy a drón szabályozáshoz szükséges utasítások az 5554-es port-on, a NavData 5555-ös port-on keresztül kommunikál a számítógéppel. A fogadó egység ezekre a funkciókra más-más értékeket alkalmaz. Most, hogy megalkottuk a szükséges csatornákat, amik a kommunikációhoz kellenek, a korábban említett repülési paraméterek felhasználásával elkészíthetjük a drón repülési konfigurációját. Ez tartalmazza az adott repülésre vonatkozó összes információt.

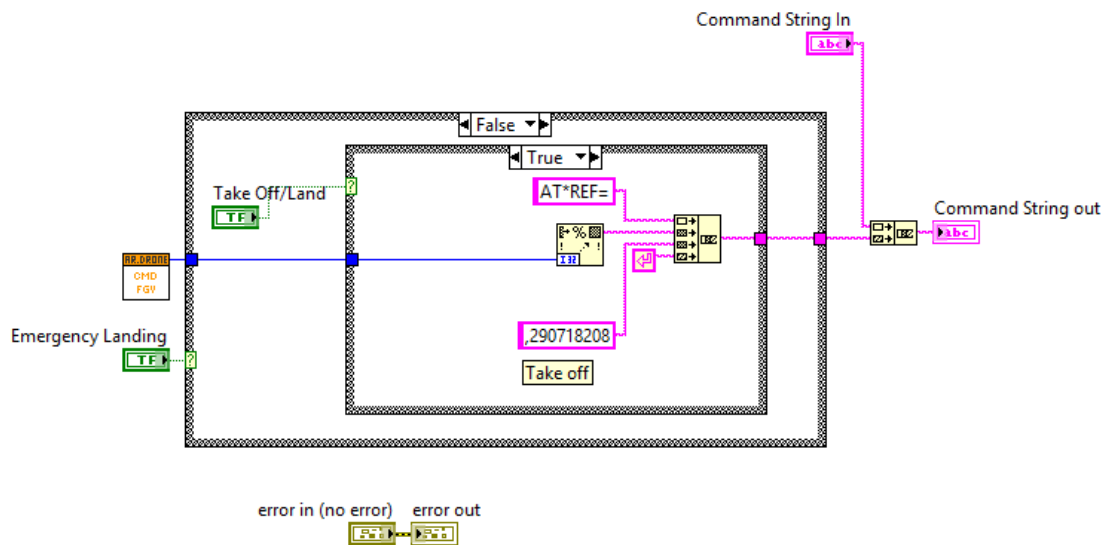
Az itt szemléltetett control elemek tartalmazzák a repülés során alkalmazott beállításokat. A listában szereplő elemek értékei kerülnek majd a konfigurációs egységbe, miután minden szükséges értéket beállítottunk. A várt konfigurációs stringet egy egyszerű tömb művelettel alkotjuk meg, és így juttatjuk el azt a rendeltetési helyére. Az előállított konfigurációs stringet újabb konkatenációs műveletek sorozatával bővítjük, így a későbbiekben már egy azonosító szekvencia értéket, valamint egy utasítás típus jelölést is tartalmazni fog. Jelen esetben ez a „AT*CONFIG=” karakterlánc lesz, amely közli a program megfelelő részeivel, hogy milyen információt hordoz az üzenet.

Fontos megemlíteni, hogy a drón ilyen folyamatosan továbbított üzenetsorozatokkal kommunikál. A konfigurációs állomány ezek után már készen áll a küldésre, amit egy Send Cmd függvény segítségével juttatunk tovább. Korábban már említettem, hogy a drón kapcsolódása során nem csak utasításokat vesz, hanem valós időben vissza is juttat videó és repülési adatokat. Ha szükséges a repülés során ezek indítása is itt történik. A 210. ábrán a hálózati kapcsolat megalkotásához szükséges inicializálási paraméterek Block diagramjának részlete látható.



21. ábra: A config string megalkotásának részlete

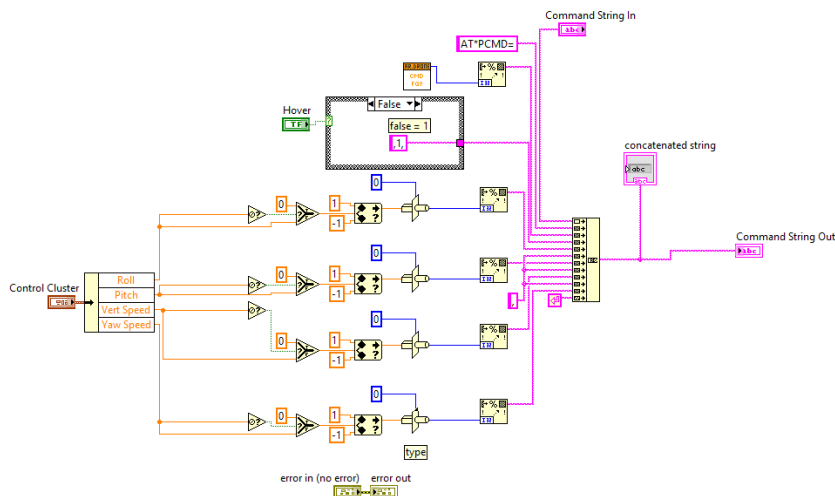
A második fázisban történik a drón szabályozására szolgáló adatcsomagok küldése. A korábban implementált hálózati kapcsolatot használva küldjük ki a drón számára az utasításokat. Két fő típust különböztetünk meg, ezek közül az egyik a referencia típusú, a másik a felhasználó által generált utasítás. A referenciautasítások alatt olyan parancsokat értünk, amelyek egyértelmű stringsorozattal leírhatók, és minden esetben azonos módon futtathatók. Ide sorolhatjuk a fel-, illetve leszállás parancsot, valamint a vészhelyzeti leállítást. Példaként, a felszállási parancsnak mindig a „AT*REF=„290718208” string felel meg, amelynek képe a 22. ábrán megtekinthető.



22. ábra: A felszállás parancs megalkotása

A felhasználói utasítások megalkotása már sokkal összetettebb feladat. Korábban már említettem, hogy a drón dinamikájából adódóan szükségünk van beavatkozási pontokra. A beavatkozáshoz szükséges irányokhoz társított utasításokat egy clusterben találhatók, amelyeket manuális esetben sliderek vagy billentyűk leütésével vezérelhetünk, a HMI szabályozás esetében pedig a kinyert adatok segítségével szabályozhatjuk a jellemzőket. A konfigurációhoz hasonlóan egy utasítás string megalkotása a cél, amelyben a megadott értékek kerülnek feldolgozásra. A szabályozás során használt utasításazonosításra szolgáló string „AT*PCMD=” karakterlánccal kezdődik. A konkatenációt követően az utasítás elküldésre kerül, amire a drón szoftvere elmozdulással reagál.

Az ilyen típusú utasítások megalkotásához, amely a 23. ábrán látható részleten is, több string összekapcsolása szükséges. A sort a szekvencia értéke nyitja, ezt követi a hover parancs kapcsolója, amely két állapotot vehet fel. Ha a helyiértékbít 0, akkor a Hover parancs aktív, és a drón csak egy adott helyben képes a mozgásra (Vertical és a Yaw), az 1-es érték esetén pedig a Roll és Pitch is végrehajtható. A mozgási formákhoz rendelt numerikus érték minden esetben -1, illetve 1 közé esik, a számreprezetáció pedig float típusú lesz.



23. ábra: A felhasználói utasítás megalkotása a mozgási formák értékeivel

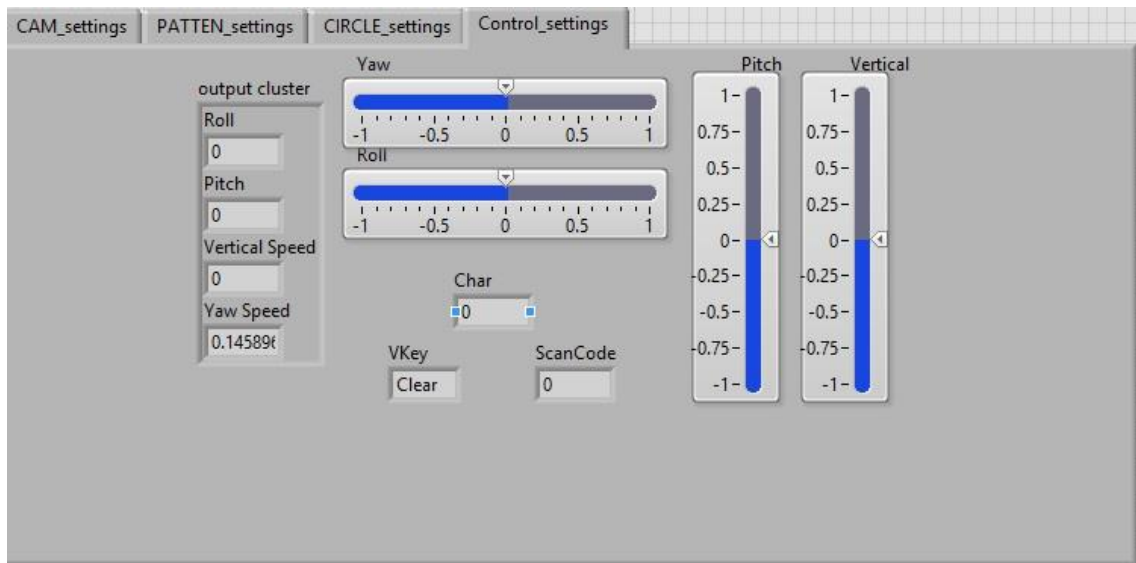
A harmadik fázisba a leszállás után juthatunk el. Ebben az almodulban a repüléshez használt és felépített hálózati kapcsolatok bontását végezzük el. Erre azért van szükség, hogy a kapcsolat során megnyitott portokat szabályszerűen le tudjuk zárni, és így nem hagyunk biztonsági kockázatot, valamint a következő felszállás alkalmával sem ütközünk inicializálási hibákba.

11. Az elkészült rendszerek összekapcsolása

A korábbi fejezetekben tárgyaltak betekintést engedtek a felhasznált megoldásokra, többek között a vezérléshez szükséges adatok kinyerésére és drón szabályozási eljárásokra. Ebben a fejezetben az eddig felvázolt elemek összekapcsolása található, valamint a dolgozat céljaként megfogalmazott feladat implementációjának bemutatása. A cél az, hogy a drónt egy webkamera és egy okostelefon által közölt adatfolyam segítségével vezéreljük.

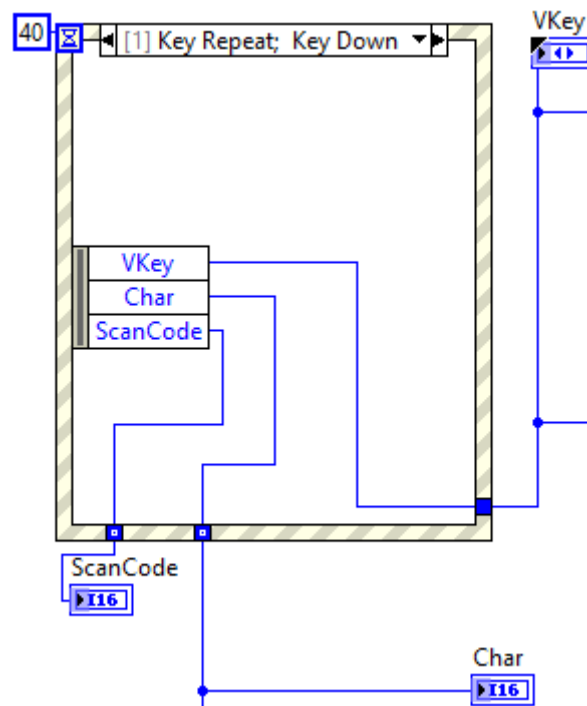
11.1. Manuális szabályozás

Az alkalmazásban, szükség esetére, helyet kapott a manuális, billentyűzettel vezérelhető szabályozó. A modul feladata, hogy a drón szabályozása akkor is megoldott legyen, ha az alany hirtelen eltávolodik, és megszűnik az adatok közlése. A választás a Front Panelen a HMI/Manual gomb lenyomásával történik. Az alábbi ábrán látható a kézi vezérléshez szükséges sliderek képe.



24. ábra: A manuális vezérlés sliderei

Erre a műveletre egy, a 25. ábrán megtekinthető, Event Structure-t alkalmaztam, aminek a segítségével figyelhetem a gomb lenyomását.



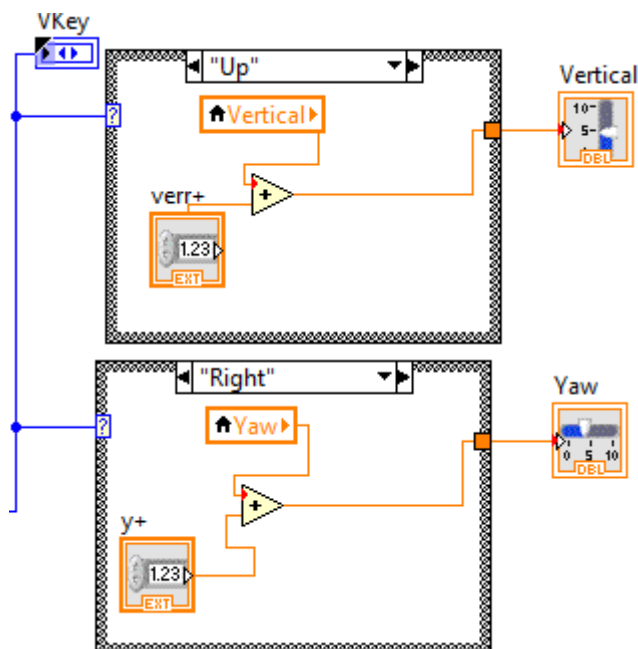
25. ábra: Az Event Structure és a szükséges paraméterek

A drón szabályozásának bemutatása során többször is említettem már, hogy 4 aktuátorral rendelkezünk, az ehhez megfeleltetett mozgási formák két irányban is értelmezettek. Például a vertikális mozgásnál a felemelkedés és leereszkedés, a yaw-nál

pedig balra, illetve jobbra forgás. Ebből jól látszik, hogy egy mozgás meghatározásához 2 darab, 4 mozgáshoz pedig 8 darab gombra van szükség.

A mozgás megvalósításához két mozgási formára használhatunk dedikált billentyűket, ezek alatt a billentyűzeten elérhető nyilakat értjük. A nyilakra a magasság változtatás és a saját tengely körüli elfordulás került kiosztásra, míg a térben történő mozgásra a több helyen is alkalmazott „w a s d” billentyűk szolgálnak. A fel és le nyilak segítségével a magasság változtatható, a balra, jobbra nyilak a saját tengely körüli elfordulást eredményezik. A kisbetűs, 'w' és az 's' karakterek, melyeknek ASCII kódja 119 valamint 115, a drón előre, valamint hátra haladásáért felelősek, az 'a' és a 'd', kódjuk 100 és 97, pedig a térben történő balra-jobbra mozgásért felelősek.

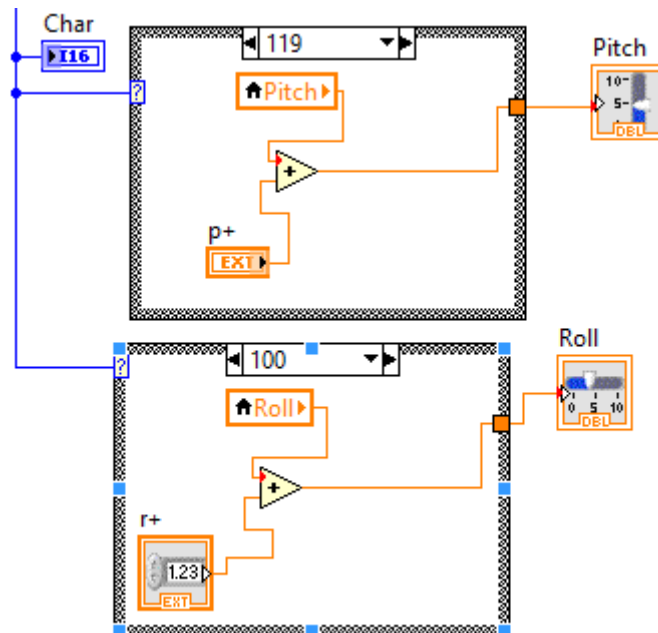
A dedikált billentyűleütéseket az Event Structure Vkey paraméterének figyelésével érem el. A 26. ábrán látható, hogy a Vkey változóban megjelenő string érték tartalmazza az irányt, amely mindkét esetben egy Case Structure-be kerül és egy Local Variable segítségével, a lenyomás hosszától függően, egy konstans értékkel módosítja az adott irányhoz rendelt értéket. A választott érték esettől függően negatív és pozitív előjellel vett 0,01 konstans erősítés.



26. ábra: A dedikált billentyűk értékátadása

A „w a s d” karakterek esetében a Char property értékének változását nézzük. A Char visszatérési értéke a lenyomott billentyű ASCII kód megfeleltetése. Ebben az esetben is

egy Case Structure segítségével reagálunk az adott billentyűleütésekre, ez látható a 27. ábrán. Minden esetben a gomb felengedését követően a változók értéke visszaáll a 0-s, üresjáratú pozícióba.



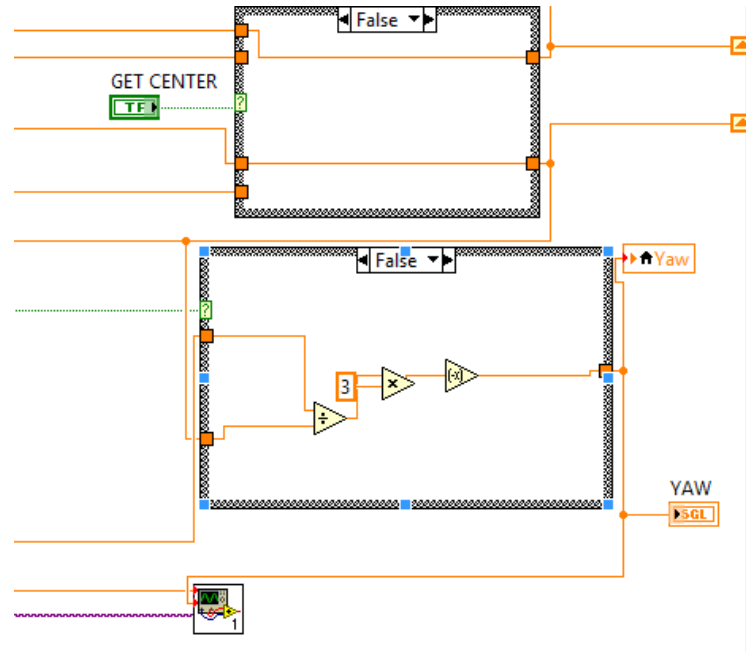
27. ábra: a „wasd” billentyűk értékátadása

11.2. A képfeldolgozásból nyert adatok

Az 6. fejezetben olvasható, hogy a pupillaközéppont megtalálást követően a középponti koordináták átadása a drónt irányító subvi-ba érkezik. Ezek az X és Y-ből alkotott pontpárok határozzák majd meg, hogy a képernyő mely pontjára tekint a vezérlő személy. A szolgáltatott adatok feldolgozása feltételhez kötött. Az adatok feldolgozása és átadása csak akkor történik meg, ha a képfeldolgozás során a találati érték nem nulla. A mérési folyamat ismertetése során említést tettem egy referencia koordináta felvételéről is, ami a pontosság és az irány meghatározására szolgál. E két pont között képzett különbséggel meghatározható, hogy melyik irányba haladjon a kopter az adott pillanatban.

A saját tengely körüli elfordulás során a rendszer megfelelő működéséhez egy ismert szabályzási eljárást választottam és alkalmaztam, ez egy proporcionális, P-szabályzóként is ismert, szabályzási módszer. Alapja, hogy az aktuálisan kapott hibajelet megszorozzuk egy konstans értékkel. A P szabályzó értékét mérések sorozatával teszteltük, és a maximális hibához állítottuk be, amelynek számszerű értéke $0,2 \frac{m}{s}$ volt, a szorzó konstans

értéke 3. A hiba értékét a referencia középpont és az aktuális pupillaközéppont között képzett különbség adta. A szabályozás a 28. ábrán tekinthető meg.



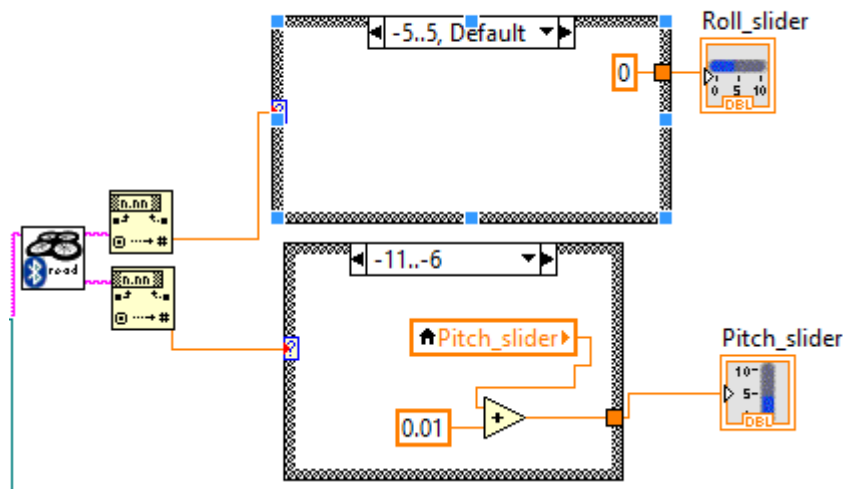
28. ábra: Az elfordulás adott erősítés mellett

A szabályozó által számolt negatív, illetve pozitív előjelű forgatási értékeket egy Property Node-on keresztül juttatjuk el a már ismert vezérlő clusterbe, amely a drón számára érthető utasítást ad az elfordulásra.

A felemelkedés, valamint a leereszkedés során a korábban megalkotott szabályozó kisebb módosítások mellett jelen esetben is ideálisnak bizonyult.

11.3. A szenzorokból nyert adatok

A szenzoradatokat a Read vi segítségével olvassuk ki. A kapott stringet a szétbontást követően egy Fract/Exp String To Number segítségével a string adattípusból egy számértéket kapunk. A kiolvasott szenzoradatokat egy Case Structure segítségével dolgozzuk fel a 29. ábrán látható módon.



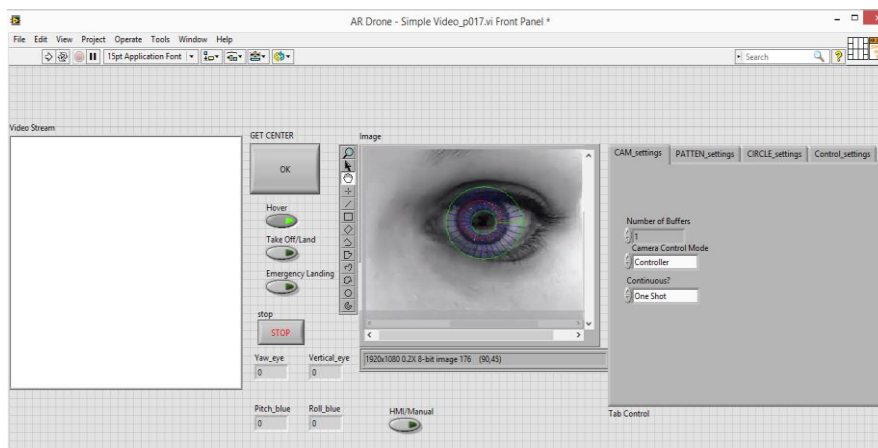
29. ábra: A szenzoradatok feldolgozása

A kiolvasott adatok csak akkor mozgatják ki a drónt a pozíciójából, ha az úgy nevezett „deadzone” határait túllépik. Az általam meghatározott érték, amin belül nincs kiadott parancs a $[-5.5]$ intervallum volt. A deadzone-on kívül eső esetben az adott irányú elmozdulási sebesség a lenyomás időtartamától függően nő, az elengedés hatására pedig megáll.

12. A főprogram bemutatása

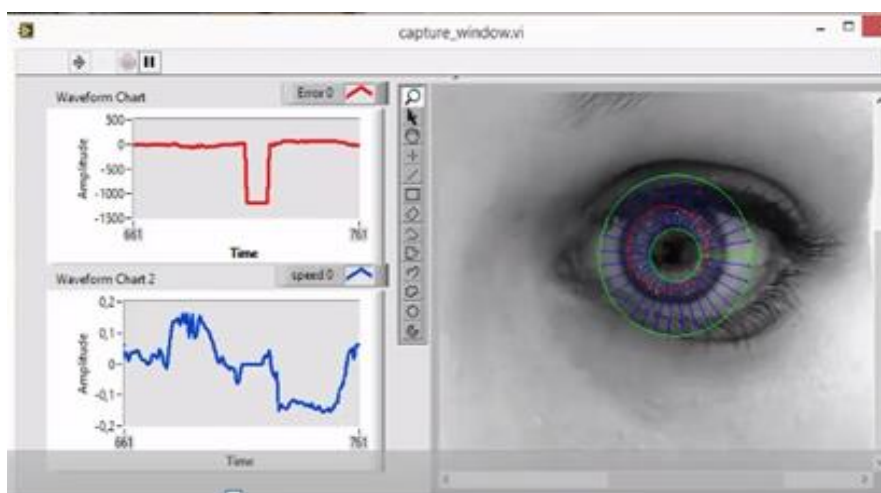
Az alkalmazás felhasználói felülete, a Front Panel, két egységből áll. Az első egység a szabályozó számítógép kijelzőjén található, a szabályozási eljárás során alkalmazott drón video adatfolyam, mellette a repüléshez elengedhetetlen referencia parancsok nyomógombjai. Mellettük található a szemmozgás követésért felelős almodul működését indikáló ablak, ahol valós időben követhetjük a szem megtalálását, és láthatjuk a találati minőséget. Ezek mellett található egy Tab Control, melynek segítségével szisztematikusan csoportokba rendezhetjük a vezérlő elemeket. A 30 ábrán látható a főprogramról készült kép.

Az elérhető Tab Control fülök elemei a következők: kamera tulajdonságok, minta illesztési eljárás paraméterei, körkeresési eljárás paraméterei, kézi vezérlés sliderei.



30. ábra: A fő program képe a szabályozó számítógépen

A főprogram második része, melynek képe a 31. ábrán látható, a másodlagos kijelzőn követhető. Ezt tekinti meg a szabályozni kívánt alany, ahol a drón valós idejű adatfolyamát láthatja, és mellette található az alany szeme, valamint a hiba és sebesség érték változások.



31. ábra: A szignifikáns adatok a másodlagos kijelzőn

13. Elkészült rendszer eredményeinek bemutatása

Az elkészült alkalmazásokkal ellátott szabályozó rendszer segítségével tesztet is végeztünk. Az 32. ábrán az élőképből kivágott képkocka látható.



32. ábra: A mérés közben készült felvétel képe

A háttérben a drón valós idejű kameraképe látható, amit a vezérlő személy is lát. A bal alsó sarokban látható a vezérlő szeme, valamint azon egy overlay segítségével megfelelően látszik a pupilla megtalálására alkalmazott algoritmus eredménye. Mellette két grafikonon, valós időben követhetőek a szignifikáns paraméterek változásai. Felül az aktuális hiba értékét indikáljuk, ez alatt pedig az aktuális elfordulás sebességének értékét láthatjuk. A mérés során több teszt is bizonyította, hogy a szabályozást végző személy képes volt a drónt a tekintetével irányítani, illetve ha egy adott pontra fókuszált, a drón lassult, majd az adott irányban lebegve ugyan arra a pontra „fókuszált”. A szabályozás megbízhatóságát növeli az is, hogy amint a mintakeresés során hiba történt, például csukott, illetve deformált szem, a drón hover parancsot adott ki, és azonnal megállt.

Az Androidos alkalmazás segítségével az alany képes volt finom mozgások megvalósítására, illetve a gyorsításnak köszönhetően nagyobb távolságok gyorsabb megtételére is. A drón ez esetben is a hover parancsot adja ki, ha nincs utasítás, mert a drón ilyen esetben a dinamikája és a keltett légáramlatok miatt elsodródna.

14. Jövőbeli tervek, elképzelések

Az aktuális fejlesztéseken és terveken kívül hosszú távú tervek is születtek. Az alkalmazás széleskörű felhasználását nagyban nehezítik a mérés során támasztott követelmények, amelyek a pontosságot hivatottak megőrizni. Számos megoldás született az álltámasz elhagyására, de ebben az esetben is a szenzorokkal történő monitorozás és korrekció tűnik a legjobb választásnak. A mérés kritikus pontját képezi a minta felvételezési eljárás, illetve az alkalmazott algoritmus sem minden esetben megbízható. Célom, hogy olyan mintát, mintabázist készítsék, amely alapján nincs szüksége a jövőben mintaképek készítésére. Távolabbi célként fogalmazódott meg a mobilitás és kényelem kérdése. Felmerült egy olyan fejre erősíthető eszköz, amely magában foglalja a kívánt szenzorokat, valamint a pupillamozgást követő kamerát és egy kijelzőt, amin látható a drón valós idejű képe.

15. Összegzés, konklúzió

A dolgozatomban egy korábbi projektben elért eredményt szerettem volna ötvözni a Műszaki Informatika Tanszéken folyó drón szabályozással kapcsolatos kutatásokkal. A szemmozgás követés során bizonyosságot nyert, hogy ez a technológia felhasználható komplexebb feladatok ellátásához is, így nagy biztonsággal alkalmazható szabályozási, irányítási feladatokra. A dolgozatomban bemutattam miként alkalmaztam a korábbi méréseim során szerzett tapasztalatot, valamint hogyan használtam fel a kapcsolódó projektem szükséges elemeit. Ismertettem a drón alapvető tulajdonságait, valamint a kommunikáció és szabályozás alapjait. Bemutattam az elkészült szabályozás eredményeit, továbbá ismertettem a jövőbeli elképzeléseket. Amint a szükséges módosítások és funkciók elérhetővé válnak, a rendszer képes lesz valós időben emberi interakcióval a drón szabályozására ezzel megalkotva a már többször is említett Human-Machine Interface-t.

16. Ábrajegyzék

1. ábra: A szemmozgás követéséhez alkalmazott mérési kör
2. ábra: A szemmozgás mozgásának útjának indikálása az idő függvényében
3. ábra: Heatmap reprezentáció
4. ábra: A drón irányításához alkalmazott mérési környezet
5. ábra: A mérőkörnyezet megvalósítása
6. ábra: A felhasznált webkamera képe
7. ábra: RGB és HSV színtartományokban látható szem
8. ábra: A pupilla megtalálása a képen
9. ábra: A szem középpontjának megtalálása
10. ábra: A fejlesztőkörnyezet képe
11. ábra: Az alkalmazás előnézete Android studioban
12. ábra: A discovery vi blokk diagramja
13. ábra: Az init vi blokk diagramja
14. ábra: A data vi blokk diagramja
15. ábra: A read vi blokk diagramja
16. ábra: A close vi blokk diagramja
17. ábra: A Parrot AR.Drone 2.0 képe
18. ábra: A drón irányításhoz szükséges aktuátorok
19. ábra: A drón és a számítógép között zajló adatok típusai
20. ábra: A drón inicializálás paraméterei
21. ábra: A config string megalkotásának részlete
22. ábra: A felszállás parancs megalkotása
23. ábra: A felhasználói utasítás megalkotása a mozgási formák értékeivel
24. ábra: A manuális vezérlés sliderei
25. ábra: Az Event Structure és a szükséges paraméterek
26. ábra: A dedikált billentyűk értékátadása
27. ábra: a „wasd” billentyűk értékátadása
28. ábra: Az elfordulás adott erősítés mellett
29. ábra: A szenzoradatok feldolgozása

30. ábra: A fő program képe a szabályozó számítógépen

31. ábra: A szignifikáns adatok a másodlagos kijelzőn

32. ábra: A mérés közben készült felvétel képe

17. Hivatkozások

- P. Müller - A. Schiffer: *Development of human eye-movement tracking system*, Ninth International PhD & DLA Symposium, 21-22 October 2013, Pecs, Hungary
- Müller P.: Szemmozgás követő rendszer fejlesztése, Szakdolgozat, PTE PMMIK, Műszaki Informatika BSc Szak, 2012, Pécs, Hungary
- Müller P.: *Az emberi tekintet követésére alkalmas rendszer megvalósítása LabView-ban*, PTE PMMIK TDK Konferencia, 2012, Pécs, Hungary
- Müller P - Schiffer A. *Szemmozgás követésére alkalmas rendszer fejlesztése*", NI Pályázat 2013, Debrecen
- Müller P.: *A Valós idejű intelligens szemmozgás követő rendszer fejlesztése*, PTE PMMIK TDK Konferencia, 2013, Pécs, Hungary
- P. Müller - A. Schiffer: *Development Quadcopter controlling by human eye-movement tracking* 10th International PhD & DLA Symposium, 20-21 October 2014, Pecs, Hungary
- P. Müller - A. Schiffer: *Development Quadcopter controlling by human eye-movement tracking*, TÁMOP-4.2.2.C-11/1/KONV-2012-0005 (Jól-lét az információs társadalomban) Closing Conference 14 November 2014, Pecs, Hungary

18. Források

- [1] <http://bigthink.com/design-for-good/the-ambulance-drone-that-could-increase-the-chance-of-survival-after-a-heart-attack-from-8-to-80> (letöltve: 2014.11.19.)
- [2] <http://www.amazon.com/b?node=8037720011> (letöltve: 2014.11.19.)
- [3] Müller Péter: Szemmozgás követő rendszer fejlesztése (szakdolgozat) 2012.
- [4] Müller Péter: NI pályázat 2013.
- [5] <http://www.zdnet.com/article/eye-tracking-web-usability> (Letöltve 2014.11.20)
- [6] <http://www.ni.com/pdf/manuals/371007b.pdf> (letöltve: 2014.11.20.)
- [7] <http://www.ni.com/pdf/manuals/371007a.pdf> (letöltve: 2014.11.20.)
- [8] <http://ardrone2.parrot.com> (letöltve: 2014.11.20.)
- [9] <https://projects.ardrone.org/projects/show/ardrone-api> (letöltve: 2014.11.20.)
- [10] <https://developer.android.com/sdk/installing/studio.html> (letöltve: 2014.11.20.)
- [11] <https://developer.android.com/samples/BluetoothLeGatt/index.html> (letöltve: 2014.11.20.)
- [12] <http://ruuhkis.com/android-repeat-action-button/> (letöltve: 2014.12.06.)
- [13] <http://ardrone2.parrot.com> (letöltve: 2014.11.20.)
- [14] <https://projects.ardrone.org/projects/show/ardrone-api> (letöltve: 2014.11.20.)
- [15] https://www.labviewhacker.com/doku.php?id=libraries:lvh_ardrone:tutorials (letöltve: 2014.10.05.)