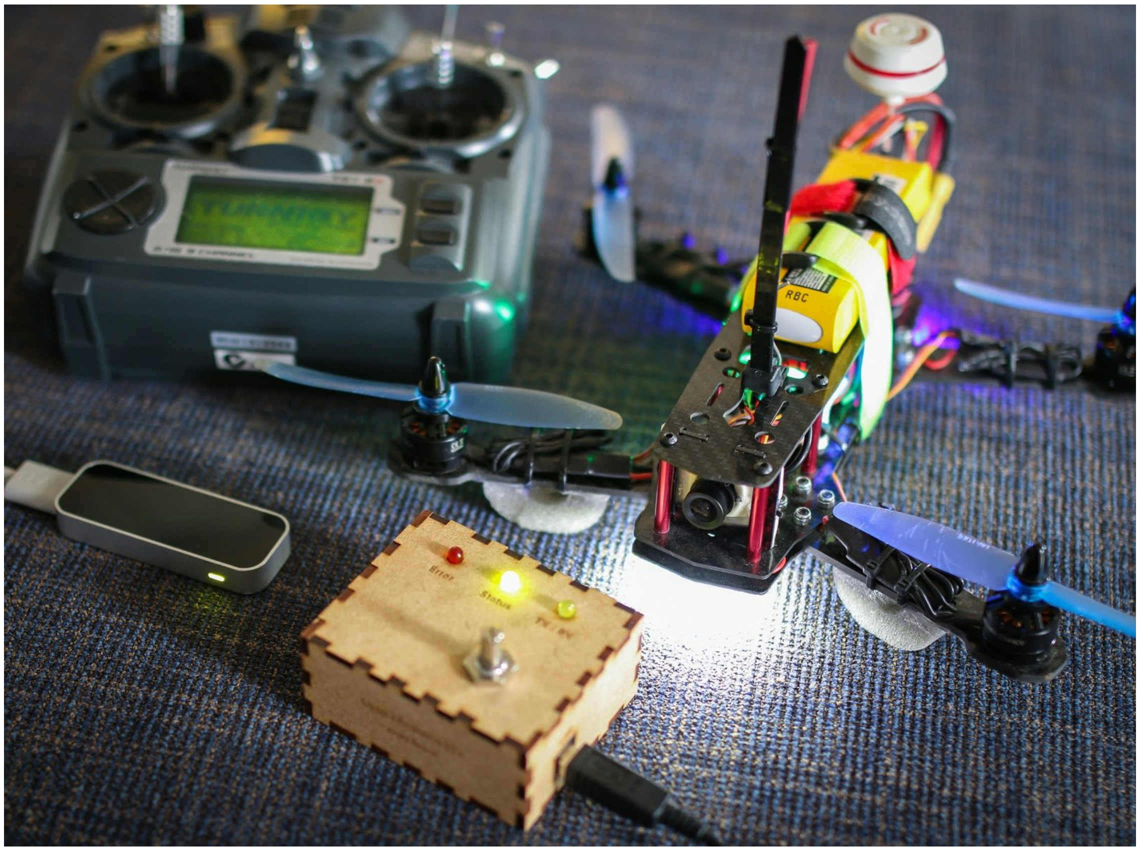
# *Digital Vision*

**VT-15**

**2015-06-08**

## Project Report: Controlling a drone using the Leap Motion



**Mentors**: *Ulrik Söderström* and *Shafiq ur Réhman*

**By**: *Arvid Bräne (arvidbrane@gmail.com).*

***Version 1.0***

## Introduction

This is a project thought of-, developed by-, and designed by Arvid Bräne. It contains some basic research on natural user interfaces and drone control with an aim to simplify the, sometimes dangerous, first time flight. The solution aims to be cheap, open-source and multiplatform requiring no/very little modification to your existing R/C-setup.

## Table of Content

## 1. Background

About two years ago I bought my first multirotor; a radio controlled little machine with four propellers connected to four electric motors. I spent about 10 000 SEK buying all the parts I needed and after that spent almost two months putting it all together to be able to fly. First flight was terrifying; controlling the *thing* with only two sticks (see Figure 1) is harder than it looks, especially since the controller was originally designed for cars, planes and boats.

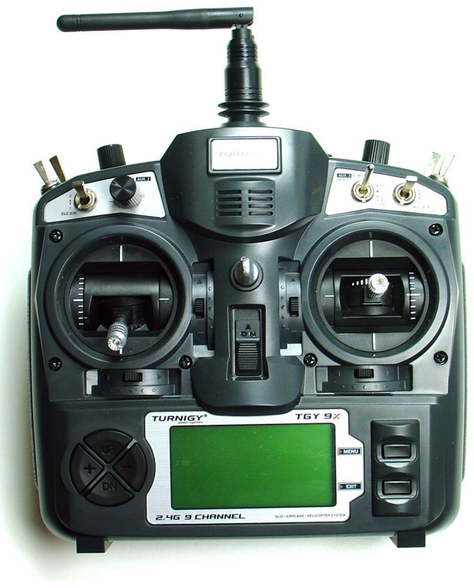


***Figure 1****: My transmitter, a Turnigy 9X*

Since that first flight I've only flown a few times, and I've crashed **<u>a lot</u>** (!). That being said; this is the most common (if not only?) way of controlling drones/multirotors today and most "pilots" start out just like me, crashing a lot and damaging a lot of expensive gear and some eventually give up due to the steep learning curve. The weird part about this is that since drones came out (about 4-5 years ago) the controller hasn't changed a bit even though they now days use them almost everywhere (especially in TV production). This is what I intended to change; I wanted to create a way for people (and companies!) to control their expensive (sometimes >200 000 SEK) multirotors that was *super simple* and intuitive and required no/very little prior knowledge of piloting. This is something that could be achieved by using a NUI [4] (Natural User Interface). Since my master in in Interaction & Design I figured this was an excellent challenge for me, some design, some programming, some testing and some building.

## 1.1 Goals

I had the following six goals (all of which have been fulfilled) for my solution in mind when this project started:

1. **Simple & Intuitive**: Easy to setup and easy to fly.
2. **Scalable**: make a modular controller system.
3. **Platform independent**: it should work on any computer.
4. **No hardware modifications**: people shouldn't have to ruin their working transmitter.
5. **Cheap**: a DIY solution that gets the job done.
6. **Documentation**: users need to be able to recreate my work.

## 1.2 Previous Work

There has not been much work done in this field before my project that I've read about, especially not with the same goals as I have had. Articles regarding controlling drones using a Microsoft Kinect sensor [2] have been out since the release of the sensor, but nothing using the Leap Motion. Some work have been done where operators are able to control a robotic arm using the Leap Motion [3] for accurate and natural control.

There is also a open-source project called "*Manucon*" (see appendix 2) which enables users to control R/C airplanes using a specially designed glove that has a potentiometer and a 9 DOF IMU sensor on top of it. In order for users to replicate they had to acquire a lot of hardware and some knowledge of electronics, something I strived to avoid. Other similar work has also been done for camera control in camera gimbals in the R/C communities, none of which work very well (see appendix 3).

Other, more scientific, progress has also been made in the fields of drones, projects like "*Sustainable Malaria-fighting Drones*" [5] and "*Delivery Drones*" [7] are really interesting, especially since drones have only been commercially available for the last few years. But there is an ongoing, and fast growing discussion, about the ethical issues about drone aircrafts [6]. Even though it is something I would like to discuss, I'm going to ignore it for this project since I don't think I should be limiting myself and the project after the ethical norms that are under discussion.

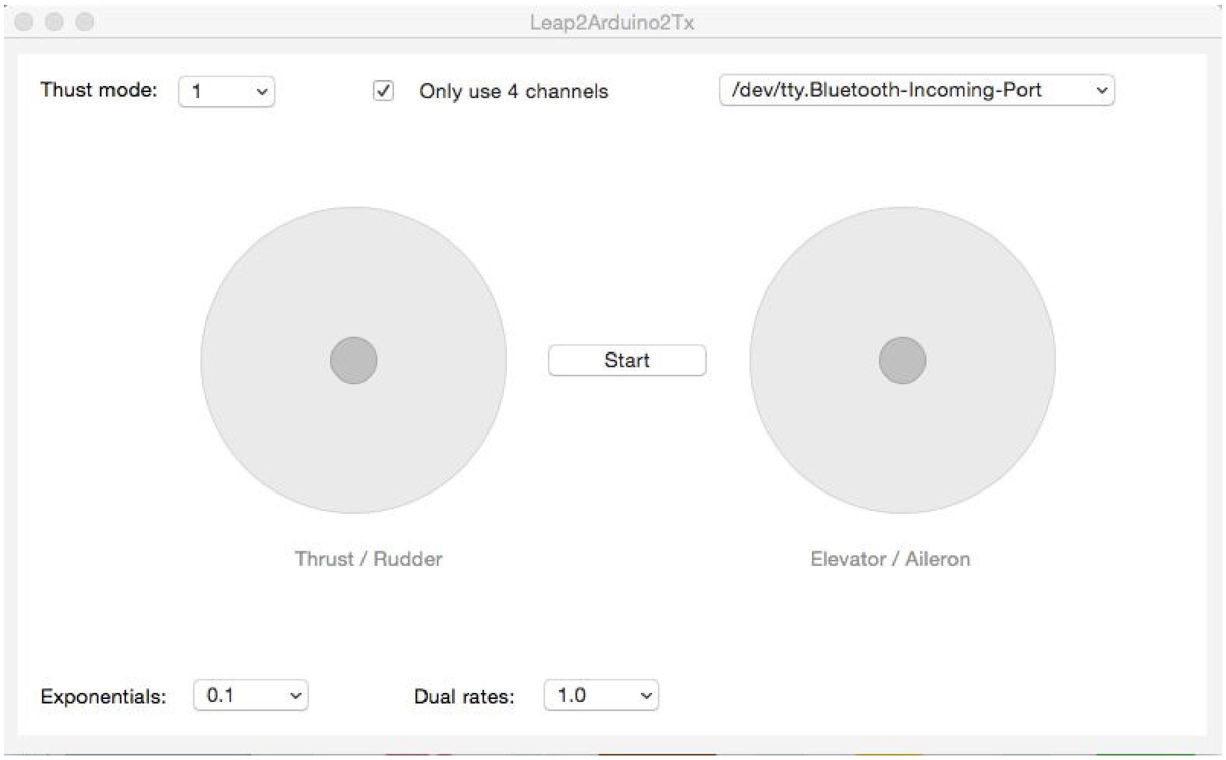
## 2. Method

During the early stages of the project I had a lot of ideas of buying a broken transmitter and scavenging it for parts and building my own hardware consisting of the R/C transmitter module, the Leap Motion, a Raspberry Pi and a battery. This solution would have been very portable and would have solved my problem (and nothing more), but it would not be very repeatable, not very cheap, not so scalable and would require a lot of hardware modifications in order to work; this was a clear "no-go".

Searching the Internet I found the DIY-headtracker project mentioned in *1.2 Previous Work* section. The goal of that project was to control a camera on an airplane in order to get different shots from the sky. Their solution was to have a basic IMU mounted on their head that was connected to an Arduino which took the data from the IMU and converted it to a PPM (Pulse-position modulation, see appendix ) signal that then was sent via a cable to the transmitter that then transmitted the data to the receiver. This solution turned out to be a good fit for my project, even though this solved something totally different using totally different controls, but it was a start.

## 3. Result

Shortly put; "*I have created an inexpensive and intuitive way to control your expensive drone with very little prior knowledge of piloting, using only standard, off-the-shelf, electronics".*



Thust mode: 1

[✓] Only use 4 channels

/dev/tty.Bluetooth-Incoming-Port

Thrust / Rudder

Elevator / Aileron

Exponentials: 0.1

Dual rates: 1.0

***Figure 2****: The Leap2Arduino2Tx GUI on the computer*

## 3.1 Solution Components

My solution is consist of six parts, one right hand, one Leap Motion (+ cables), one Computer (with Python installed), one arduino (+ cables), one R/C transmitter, and one Drone (or some other R/C vehicle), see *Figure 3*.

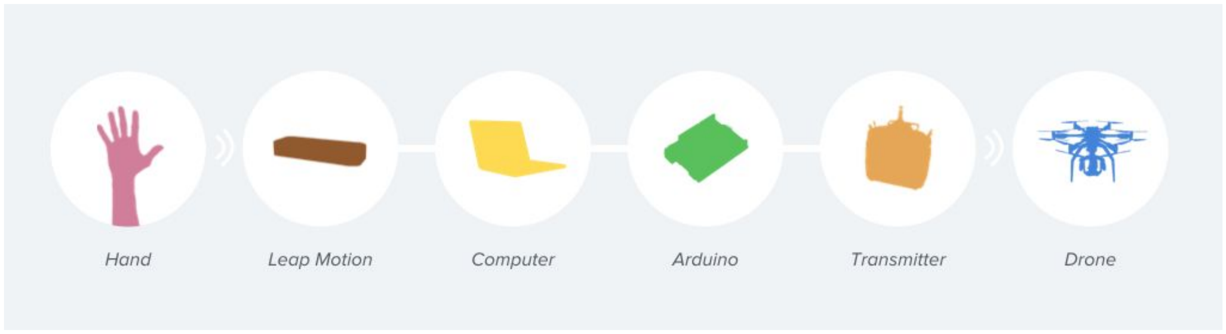


***Figure 3****: My 6-part solution*

### 3.1.1 Hand

Almost everyone is born with two hands and as we grow up we learn to control them with very high precision and next to zero errors. My solution uses the motion of your right (no left hand support at the moment) hand to determine how/where to steer the drone. If you yaw your hand right, the drone will yaw right; if you pitch your hand back, your drone will pitch back, etc. To get at better understanding of how this work I recommend taking a look at appendix 5.

### 3.1.2 Leap Motion

The Leap Motion uses IR-lights to get a 3D depth map of your hand at about 300 times per second with a very high accuracy [1]. Using the data that the sensor acquires from scanning the hand the Leap figures out which part of the hand is which and sends it over to the computer via USB.

### 3.1.3 Computer

The values are then received (about 20 times per second) calculated, translated, mapped, checked and visualized (see *3.2 Technical Part*) before they are sent, over USB, to the Arduino. This is also where the user gets the feedback from his/hers input (see Figure 2) and make all the configurations before the flight.

### 3.1.4 Arduino

The values from the computer are received and then converted to a PPM signal with very low noise. This signal is then sent, about 50 times per second to the transmitter via a 3.5 mm cable plugged into the trainer port on the transmitter.

### 3.1.5 Transmitter

Since the data is coming in via the trainer port, the transmitter believes that it is connected to another transmitter (of the same- or similar model) while it’s really connected to Arduino emulation to be another transmitter. This allows the for the values to be sent, just like normal, to the drone.

### 3.1.6 Drone

The drone on the other hand has no idea who or what that is controlling it, it only cares about the values that are being transmitted and acts accordingly.

## 3.2 Technical Part

There are really only two parts that need a technical explanation and those are the Computer-side and the Arduino-side. For a better, more commented, explanation I recommend having a look at the source code (see appendix 1).

### 3.2.1 Computer

The following transmitting process happens in this chronological order in every loop inside the Python-program on one dedicated thread:

1. The values (hand-height, hand-roll, hand-yaw and hand-pitch) are fetched from the Leap
2. The values are remapped from their different scales (height, radians etc.) to -1 to 1.
3. Exponential conversions are then, if any, performed (see appendix 7).
4. Dual rate endpoints are then, if any, applied (see appendix 7).
5. Values are then converted (remapped) into channel values (ranging from 113 to 937).
6. Channel values are than transmitted in chronological order in CSV-format over USB.

This whole process happens about 8 times per second on my five year-old Macbook Pro, but could very well run faster on a computer with better performance. It should also be mentioned that the program runs a separate thread for the GUI updates to maintain the performance on the transmitting thread.

### 3.2.2 Arduino

The following happens in chronological order in every loop inside the C-compiled-program:

1. Parse new values every time a transmission has been received.
2. Apply the newly parsed values to the channel-value list.

Along side this loop the program has a timed function that triggers every 22 milliseconds and there the following happens in order to create a PPM signal (see appendix 8):

1. For every channel in the list (in my case I have a 8 channel transmitter):
    a. Transmit 0 for 300 microseconds (channel value separator)
    b. Transmit 1 for 650 microseconds (minimum value for each channel)
    c. Continue transmitting 1 for the number of microseconds of the current channel
2. Transmit 0 for 300 microseconds (channel value separator)
3. Transmit 1 until the next function call (synchro time pulse)

## 4. Experiments

To perform scientific test and experiments for this project is hard due to two major reasons; the fact that flying is hard and bad weather. When my solution was working fairly stable I decided to give it my first try indoors; the maiden flight was terrible and it was very hard to control. My initial idea about thrust control (mode 2, see Control section in appendix 1) turned out to be really hard to hit the sweet spot; the drone keep increasing/decreasing thrust and it was too hard to hoover. After this test (and two more just like it) I went back to the drawing board and added another way of thrust control (mode 1, see Control section in appendix 1) which turned out, in about test 6 or 7, worked much better. During the last week of the project I also added a third way of thrust control (mode 3, see Control section in appendix 1) using a separate potentiometer connected to the Arduino. Unfortunately I never got around to try this solution.

### 4.1 Results

Due to the complex way of connecting all the components and understanding the arming/tracking sequence during the early stages of the project I only got around to do test with three other pilots than me during the last week of the course, when my solution was more neatly packaged. These test where performed indoors, due to poor weather conditions (rain and heavy wind), with first time pilots with no prior knowledge of the R/C hobby. The results are a bit fuzzy, especially since I didn't take any notes during the interviews, but the content of all three was something like this: "*Its not easy to control, but it's easier than the normal transmitter*". While this is not really the result and reactions I was hoping for it is a step in the right direction, especially since I spent about 3 hours in flight simulators before I tried, and failed, my first flight using my transmitter.

## 5. Discussion

Here are some of my thoughts on this project.

### 5.1 Problems

There are a couple of problems that need to be solved in order for this to be a reliable solution, here are some of the most important ones:

1. Reduce the latency, currently the USB transmission from the computer to the Arduino is a major bottleneck since the board can't handle that much data at once.
2. Fix the occasional UI-freezes (don't worry, the app is still transmitting the values and your hand is still being tracked) in the GUI.
3. Try it on a more sophisticated and more reliable platform; I used a home built racing quad that is built for speed, not for usability since it has a very limited number of sensors.
4. More reliable failsafes; something that can only be done by solving the problem above.
5.

### 5.2 Future

There is a lot of room for improvement (see *5.1 Problems*), but when those things are solved I would like to expand this project sometime in the future; here are some ideas I have:

1. Add more ways of controlling, maybe a Joystick or a simple Keyboard.
2. Add Cameraman-control/mode.
3. Add lefthand control.
4. Take the Arduino auto of the equation and use the AUX port on the computer instead.

## 6. Conclusion

While the results are not as clear as I wanted them to be and the solution isn't as efficient as I accepted it to be I'm still very happy with what I have accomplished. I have learned a lot and lots to show for it.

Half way in during the project I sent an email to DJI (see appendix 9) asking them to sponsor me with one drone (since their drones are very capable of automated flight due to their large quantity of on-board sensors), the Phantom 3 or 2 to be precise. Unfortunately I didn't receive an answer (I didn't expect one, but I really hoped for a reply back...) and soon I forgot about it, but a week after the final presentation of the project a video (see appendix 10) popped up of DJI showing of them controlling one of their drones using a Leap Motion. Coincidence? I think not; the Leap Motion has been out for about two years and DJI has been making drones for over five years.

## 7. References

1. Marin, G., Dominio, F., & Zanuttigh, P. (2014, October). *"Hand gesture recognition with leap motion and kinect devices"*. In Image Processing (ICIP), 2014 IEEE International Conference on (pp. 1565-1569). IEEE.
2. *"Altitude control of a quadrotor helicopter using depth map from Microsoft Kinect sensor"*, ISBN: 978-1-61284-982-9, Stowers, J. ; Electr. & Comput. Eng., Univ. of Canterbury, Christchurch, New Zealand ; Hayes, M. ; Bainbridge-Smith, A.
3. "*Intuitive and Adaptive Robotic Arm Manipulation using the Leap Motion Controller*", ISBN: 978-3-8007-3601-0, Bassily, D. ; Georgoulas, C. ; Guettler, J. ; Linner, T.
4. *"The future of natural user interfaces*", ISBN: 978-1-4503-0268-5, Jhilmil Jain, Arnold Lund, Dennis Wixon.
5. "*MedizDroids Project: Ultra-low cost, low-altitude, affordable and sustainable UAV multicopter drones for mosquito vector control in malaria disease management*", INSPEC: 14817478, Phelps, D. ; Oladipo, O. ; Sewovoe-Ekuoe, F. ; Jadoonanan, S. ; Jadoonanan, S. ; Tabassum, T. ; Gnabode, S. ; Sherpa, T.D. ; Falzone, M. ; Hossain, A. ; Kublal, A.
6. "*Ethical issues with use of Drone aircraft*", INSPEC: 14581412, Wilson, R.L.
7. "*Can Drones Deliver?*", IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, VOL. 11, NO. 3, JULY 2014.

## 8. Appendices

1. **The source code**: https://github.com/Kodagrux/Leap2Arduino2Tx

2. **Manucon**: https://github.com/RaptorTech/Manucon
3. **DIY headtracker**: http://www.rcgroups.com/forums/showpost.php?p=21974105
4. **Leap2Arduino2Tx Setup video**: https://www.youtube.com/watch?v=faQnHgn8Z-0
5. **Leap2Arduino2Tx Contol video**: https://www.youtube.com/watch?v=iUCDGIBrwPQ
6. **A blog-post about Leap2Arduino2Tx** (coming late June): http://www.arvidbrane.se/blog
7. **Video explaining expo & dual rates**: https://www.youtube.com/watch?v=skU51E5ilG0
8. **PPM-signal explanation image**: http://bit.ly/1MoK00L
9. **DJI's website**: http://www.dji.com
10. **DJI's Leap-video**: https://www.youtube.com/watch?v=wFo21xrFGr0&t=404