The force is with you: Controlling autonomous bots with a wave of your hand

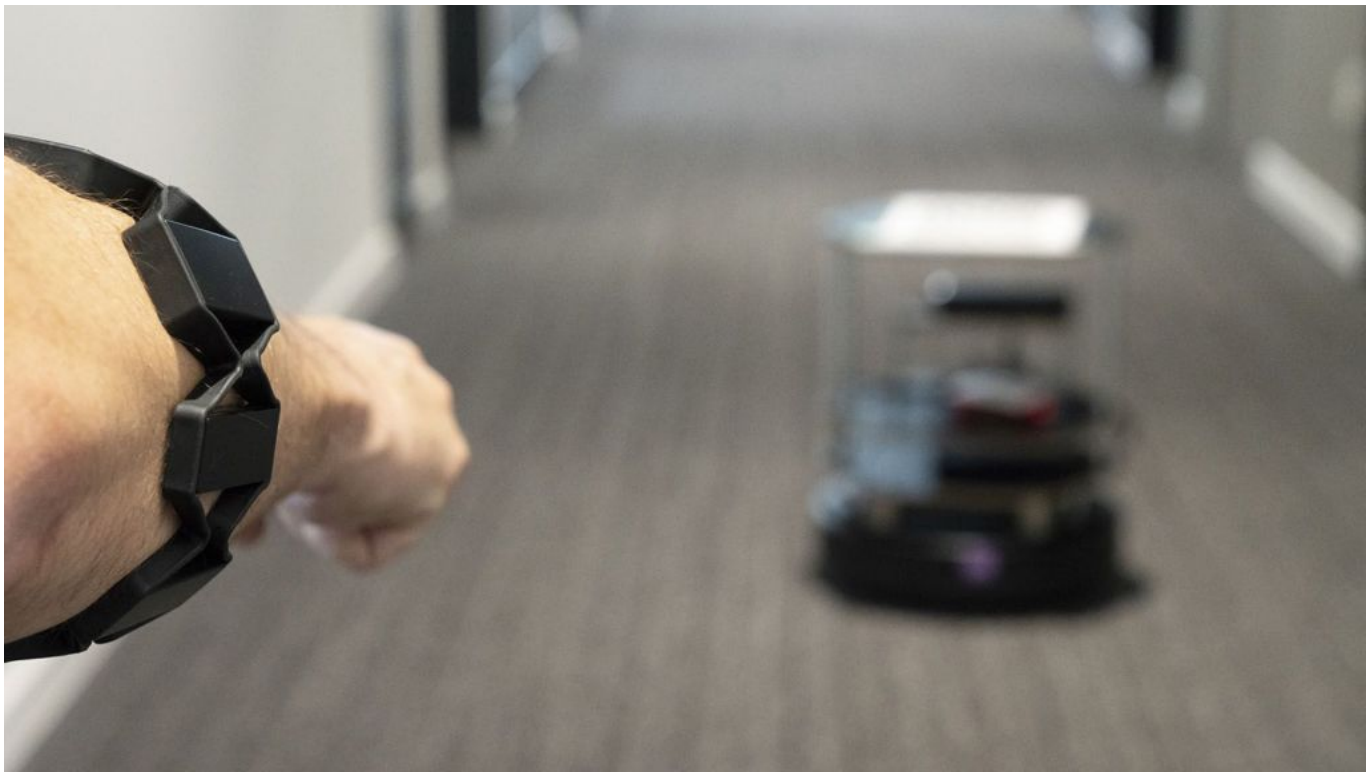
28 June 2018 | **Augmented & Virtual Reality, Insight Zuhlke** | **Michael Sattler**
Reading time: 7 minutes

Who wouldn't want to feel like Darth Vader, moving objects around just by pointing your arm at them while wearing a fancy headdress? For our Zühlke Camp, this sounded like exactly the right project. As obviously we were lacking the "real Force", we needed some technical gadgets to help us out: A Microsoft HoloLens, a Myo gesture control wristband and a TurtleBot. After just three intense days, we were ready to show off our results – and feel a little like actual Jedi (or former Jedi, for that matter).

But let's get back to the start: In Mixed Reality on Microsoft HoloLens, gesture control using the direction of your gaze and the Air-Tap gesture has become a well-accepted pattern for interacting with apps. By holding the Air-Tap gesture closed, you can not only manipulate and move virtual objects by moving your head but also by moving your hand and "dragging" objects. However, this manipulation scheme has its limitations: When your hand moves out of the HoloLens' field of view, it can no longer be tracked. Also, the Lens can only detect where your hand is (roughly), but not how it's oriented. And having just one gesture available might also not be enough in some cases.

An Armband for the Augmented Reality

Enter Myo, the gesture control armband by Thalmic Labs! Worn on the forearm, it can track the movement of your arm, regardless of where you're moving it. It can also track the movement of your fingers using myoelectric sensors, making it possible to detect five discrete hand gestures. With its Bluetooth Low Energy (BLE) interface, it can easily connect to a wide range of devices.



At this year's Zühlke Camp we did a number of Augmented Reality experiments, including a small showcase that enables users to control the movement of a TurtleBot with a Myo connected to HoloLens. The idea was to not just use gestures to trigger forward/backward movement and rotation, but to guide the bot by pointing to a location where you want it to go.

Interacting with Myo

At first, we had to establish the connection between HoloLens and Myo. While the regular release version of Windows 10 on HoloLens didn't properly support BLE prior to June 26, the new RS4 Release (which is officially available now) works nicely here: BLE peripherals can be detected, connected, and access to all GATT characteristics works without any limitations. Documentation on Myo's BLE API is available in the form of a C++ header file. With this header and the code available in the Myo example projects for Unity, we were able to write a wrapper around the BLE interface to easily connect to a paired Myo and comfortably use all of its features in a HoloLens app.
The feature that was easiest to use on Myo were hand gestures. In the BLE interface, whenever your hand pose changes to one of the recognizable gestures (or back to a resting position), you get a GATT Indication.

Gesture Control à la "Minority Report"

We found Myo's arm orientation tracking to be very fast, precise and stable, allowing us to

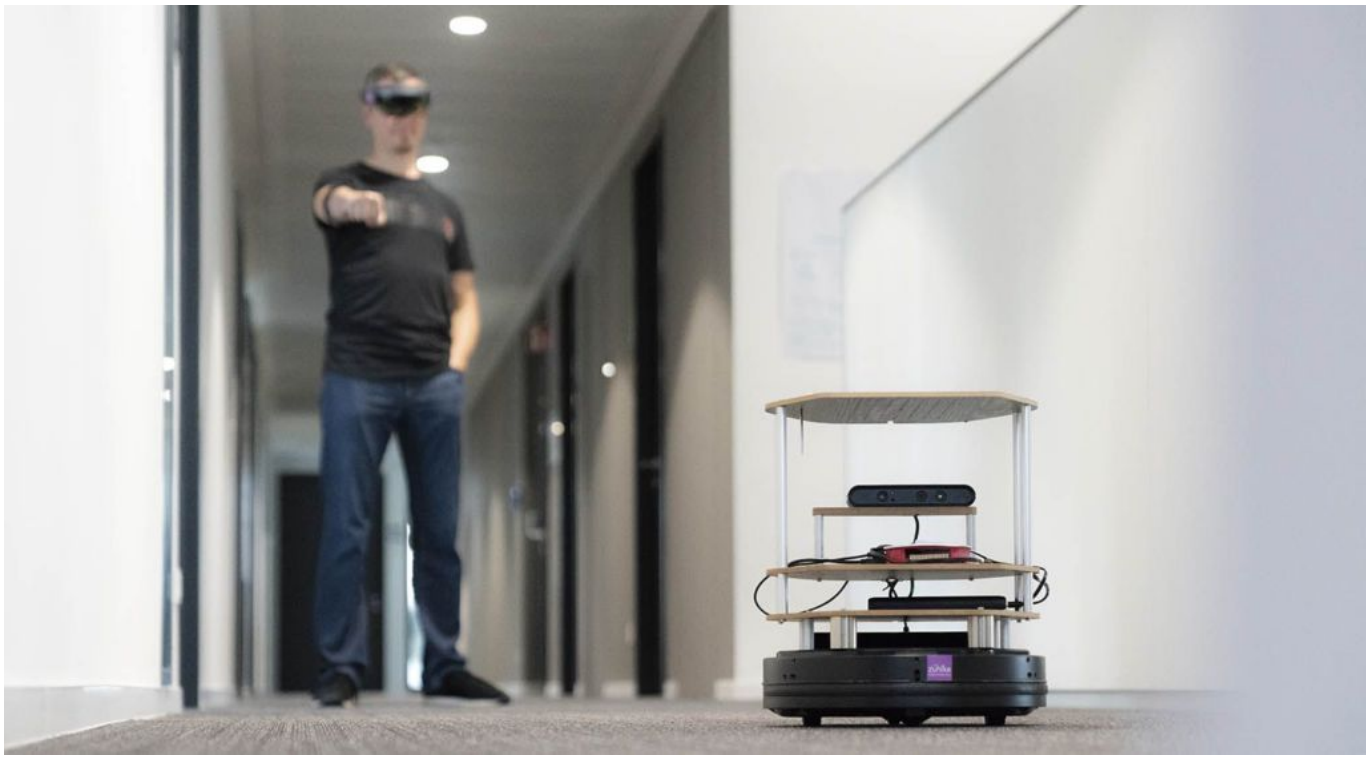
map the direction of the user's arm to an object inside our app in real-time. This was also pretty straightforward: The BLE interface frequently sends GATT Indications that contain the current rotation which in turn can be applied to an object's transform in your scene. We just had to make sure that the initial detected direction was in sync with our app's coordinate system. To achieve this, we had the user point his arm in a direction suggested in the app and confirm with a gesture.

The feature that was most difficult to use was Myo's accelerometer. Myo sends precise acceleration values on all three axes, but you have to factor out earth's gravity as well as any possible calibration error. This made it hard for us to use the accelerometer values for keeping track of Myo's position in space. We found that it made more sense to detect patterns in the motion of the user's arm to use them as additional gestures on top of the five hand gestures. Although we didn't use this feature for our showcase in the end, it's easy to see how this can be used to implement cool arm gestures like in the movie "Minority Report".

One thing that frequently tripped us was the fact that some of Myo's gestures can potentially be recognized as HoloLens gestures as well. For example, the "Finger Spread" gesture can easily trigger the "Bloom" gesture on HoloLens, closing your app. This was especially nasty with the HoloLens RS4 preview OS where gesture recognition is a lot more sensitive. We had to take this into account when we chose which of Myo's gestures to use in what situations.

Controlling the TurtleBot

With Myo connected to HoloLens and our app receiving all the input we wanted, the next building block was to navigate the bot to the location marked by the user. At first, we wanted to use the TurtleBot's spatial awareness to implement the necessary navigation routines directly on the bot. However, given the short time frame for the implementation (we had only three days to get everything working), we instead implemented the navigation routines directly on HoloLens.



To detect the position of the Bot in the room, we simply taped a visual marker onto its top deck. By tracking the position and orientation of the marker, we could automatically calculate the corresponding movement commands for the Bot to guide it to its destination.

The TurtleBot uses MQTT as its communication protocol. The endpoint can be reached from components deployed on the bot or from remote devices on the same Wi-Fi, such as the HoloLens. The topics on the MQTT interface can be read from or written to. MQTT topics are linked to topics in ROS, which powers the TurtleBot. The ROS topics can in turn be used to access the various features of the TurtleBot. On HoloLens, we implemented an MQTT client using uPLibrary to send linear and angular speed parameters to the bot.

Combined with the bot's position as tracked via the visual marker, this gave us everything we needed to implement a basic navigation mechanism to send the bot to any location on the floor that we pointed our arm at. Of course, having to keep the bot in sight of the HoloLens camera to track it was a little cumbersome, but it worked! For three days of coding with four people on our team, this was a great achievement – and it was also a lot of fun seeing everything come together.

The future

Of course, what we achieved was great, but most important were the things we learned. For instance, the power put in your hands with Myo is a lot greater than what was obvious at first. The combination of an array of available gestures with stable and precise arm tracking

gives you a lot of possibilities when combined with Mixed Reality. This became especially clear when we showed the results to our colleagues and talked about the showcase and the hardware involved. There were lots of ideas for Myo in the context of other existing projects and also many ideas for entirely new project setups. Also, the prospect of being able to use two Myos on your arms led to a lot of fantastic thoughts on what could be achieved. So it was absolutely clear that we'd have to flesh out the connection between HoloLens and Myo even further.

The same goes for TurtleBot: The platform has so many possibilities and interfaces that it can be used for countless experiments and scenarios, inside and outside of Mixed Reality. Think about its sensor array, for example: A TurtleBot could automatically map the geometry of a room, send it to a HoloLens before the wearer even enters the room and then carry out tasks in the room autonomously as commanded from HoloLens.
We can't wait to do some more experiments! With tools like these, combined in a clever way, you feel like the Force is right in your hands!