Our annual educational “Camp” at Zühlke Germany was a real blast – in several regards! For one, the location at Castelldefels near Barcelona provided a really beautiful and relaxing environment. But the educational aspect was great fun as well, as this year a few colleagues and me dug into Augmented Reality. We wanted to get to know the intricacies of Vuforia and Unity in order to use this combination for rapidly prototyping Augmented Reality applications.

We wanted to prototype a solution for a common problem in hardware development: Once an electronics design has been manufactured, it does not behave as expected. The analysis is very time consuming and involves studying the blueprint and doing some calculations by hand.

We envisioned a solution, where the board is simulated and an AR app is used to facilitate checking probe points quickly to locate the source of the problem.

For our four-day camp we stripped it down to a simple showcase: we were aiming to “augment” an STM32F0 development board with information on the pin assignment and status.

**The Setup**

Before we took off for the camp, we already did some preparations for our project. To augment the board, we needed to set up an augmentation target for Vuforia to recognize. We tried out two approaches here: Using a simple bird’s eye view snapshot of the board as a so-called Image Target, and creating a 3D Object Target using Vuforia’s Android Object Scanner app. We tried a simple augmentation with these two approaches, and both worked. On the smartphone side, we chose to go with Android because we wanted to use Bluetooth Classic. We rigged up the STM32 with a simple Bluetooth module for communication with the app. The basics for the embedded software were written in advance as well.

**The Approach**

We agreed on the following approach for conveying the needed information in the UI: We render a “crosshair” in the middle of the live camera image that the user places over any pin, LED, or button by pointing the camera. The element in the middle of the crosshair is highlighted by placing a rendered cylinder on top of the element. When an element is
focused like this, any status information available is displayed in a “Head-Up Display” at the top of the screen.

For communication between the STM32 and the app, we’d have to use a plugin, as communication via Bluetooth is not possible “out of the box” using scripting in Unity. We went with “Android Bluetooth Multiplayer” from the Unity Asset Store. In hindsight, however, the plugin “Android & Microcontrollers” would probably have been a better choice in our case, as explained below.

**The Implementation**

We started out by first implementing basic augmentation of the STM32 board to make sure our environment worked properly. To save time, we initially used the camera on our development laptops instead of an Android phone for this. As soon as we saw that working, we started setting up our Unity scripts in C# to breathe some life into our Augmented Reality. The first challenge was finding out the pin or LED the user is looking at. We found that to accomplish this, a Vuforia Image Target helped a lot. Unlike an Object Target, the Image Target is represented by a visual object in the unity scene that has the exact proportions of the actual target. This visual object can be used for a Raycast operation in Unity’s physics engine to pinpoint the location on the target that is hit by the “ray” in the center of the view. The point could then be mapped to the nearest pin or LED to mark it “selected”.

After finding out the selected element, there were two things to do: We first placed the highlight cylinder on the element, modified to approximate the dimension of the given element. Then we queried the Bluetooth interface for status information on the selected element. Any status information available would then be displayed in the HUD, which was created using a simple Unity Canvas.

The Bluetooth interaction using the “Bluetooth Multiplayer” plugin was a little tough to do, as this plugin is actually designed for communication between two Unity clients. To talk with the STM32, we had to bypass some of the plugin’s features and implement the communication in JNI. Needless to say, this was extremely cumbersome. Moreover, any use of the System.Threading API for the Bluetooth communication led to crashing the app. Luckily for us, the communication was fast enough to query for status information in every frame without causing noticeable lag.

The Bluetooth communication needed to be implemented on the embedded side as well, though we were facing less difficulties here.
After the app ran smoothly on Android phones, we also made it work with Google Cardboard. To achieve this, we just needed to switch Vuforia’s AR Camera in the Unity scene to “Video See-Through Eyewear” and set up a HUD Canvas for the left and right “eye” of the AR Camera. With part of the Cardboard’s front cut out for the smartphone camera, we now had a nice hands-free goggles version of our Augmented Development Board utility.

**The Lessons Learned**

It was nice to see how smoothly Vuforia and Unity work together to quickly build Augmented Reality applications. The APIs and tools are easy to use and well documented, making it easy to get into them and make good progress very fast. Within just three days, our team of four developers from very diverse disciplines (who delved into these tools for the first time) was able to create a nice AR showcase with practical use. This showcase has immediately gained the attention of many colleagues, management and even some potential customers.

We are looking forward to any projects in the field of Augmented Reality in the future! Stay tuned, we have some big announcements in store!