

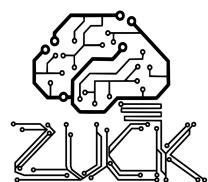
# **Projektbericht ZUCK**

Zentrale Unterhaltungs- und  
Consumerprodukte-Kontrolleinheit

Betreuer Professor:  
Prof. Dr.-Ing. David Strippgen

**TOBIAS MAYR**

HTW Berlin  
Internationale Medieninformatik  
SS 2017



Home Automation Server ZUCK (Zentrale  
Unterhaltungs- und Consumerprodukte-  
Kontrolleinheit)

## Inhalt

<b>1 Die Idee</b>	<b>3</b>
<b>2 Planung</b>	<b>4</b>
<b>3 Organisation</b>	<b>4</b>
<b>4 Aufgabenverteilung</b>	<b>4</b>
<b>5 Projektmanagement</b>	<b>5</b>
<b>6 Hardware setup</b>	<b>6</b>
<b>7 Server Setup</b>	<b>7</b>
<b>8 Softwareentwicklung</b>	<b>8</b>
<b>9 Technische Hürden</b>	<b>10</b>
<b>10 Showtime</b>	<b>13</b>
<b>11 Fazit und Ausblick</b>	<b>14</b>

## 1 Die Idee

Am Tag der Projektvergabe hatte ich das Glück in das von mir favorisierte Projekt eingeteilt zu werden. Die Vorgabe war ein Home Automation System zu konzipieren und zu verwirklichen welches IoT-Geräte steuert aber auch analoge Haushaltsgeräte über beispielsweise Funk oder Infrarot kontrollieren kann. Als zentrale Servereinheit soll ein NanoPi NEO Air benutzt werden welcher auf Linux/NodeJS basiert. Des Weiteren werden mehrere ESP8266 Nodes benutzt um proprietäre Systeme anzusteuern. Unsere Aufgabe war es nun uns zu entscheiden welche Geräte wir in das System integrieren wollen und eine Infrastruktur zu entwerfen welche sowohl Hardware als auch Software einbezieht. Das System sollte also möglichst viele interessante Features bieten und gleichzeitig sehr preiswert sein durch die Verwendung von erschwinglicher Hardware.

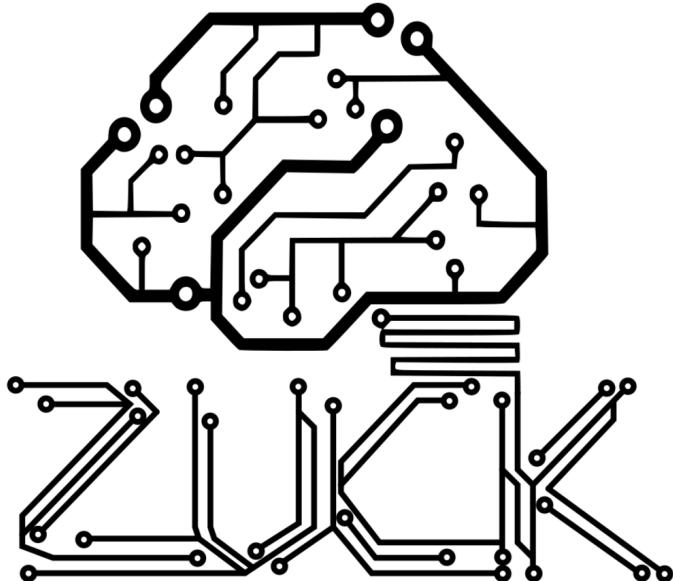


Figure 1: ZUCK-Logo

## 2 Planung

Anfangs haben wir uns geeinigt uns einmal wöchentlich zu treffen um erst einmal Ideen zusammen zu tragen, uns in Node JS einzuarbeiten und uns mit der Hardware zu beschäftigen. Nach dem ersten Treffen hatten wir eine längere Liste von Ideen gesammelt welche wir dann priorisierten und die besten ausgewählt haben um sie umzusetzen. Darunter waren Ideen wie beispielsweise einen Feuchtigkeitssensor für Topfpflanzen zu integrieren um benachrichtigt zu werden wenn die Pflanzenerde zu trocken ist oder einen Smart-Mirror welcher Statusnachrichten anzeigen kann. Diese beiden Punkte haben wir im Verlauf des Semesters in die Tat umgesetzt aber es gab auch Ideen die wir anfangs integrieren wollten aber uns dann doch umentschieden haben. Dies hatte entweder damit zu tun dass wir mehr Zeit und Aufwand in andere Funktionalitäten investieren wollten oder wir die Idee nicht für gut genug hielten um sie in das System einzubeziehen. Dazu gehört zum Beispiel das einbeziehen von Emails in der Statusanzeige und Überwachungskameras welche das Eigenheim kontinuierlich filmen und den Nutzer über Benachrichtigungen informieren kann falls das System Bewegungen aufzeichnet. Letzteres wurde unter anderem auch aus der Liste geworfen da wir Bedenken über die Sicherheit des Systems hatten da sich ein Angreifer Zugriff auf die Kamera verschaffen kann und somit die Privatsphäre des Nutzers stark gefährden kann.

## 3 Organisation

Bis etwa zur Hälfte des Semesters wurden die wöchentlichen Treffen beibehalten. In der zweiten Hälfte häuften sich unsere Termine von zwei Terminen bis hin zu fast täglichen Treffen im letzten Monat vor der Showtime. Zur Kommunikation benutzten wir WhatsApp und SMS-Nachrichten. Gearbeitet wurde fast ausschließlich an der Hochschule aber auch alleine oder in der Gruppe zuhause. Für die Dokumentation nutzten wir das in Redmine integrierte Wiki in dem wir Tutorials sammelten und wichtige Arbeitsvorgänge niederschrieben um die erforderlichen Schritte für die Einrichtung und Erweiterung des Systems festzuhalten. Als Ticketsystem und um unseren Fortschritt zu "tracken" haben wir ebenfalls Redmine benutzt. Zur Versionsverwaltung kam Git zum Einsatz. Anfangs haben wir das Repository auf dem Hochschulserver abgelegt sind aber auf den webbasierten Hosting Dienst "GitHub" umgestiegen da die Weboberfläche mehr Möglichkeiten bietet.

## 4 Aufgabenverteilung

In den ersten Treffen wurden die Aufgabenbereiche zugeteilt anhand von persönlichen Interessen und Fähigkeiten. Ich wollte gerne Node JS lernen und habe mich demnach im Verlauf des Semesters besonders stark mit Node

JS auseinandergesetzt und mich auf die Entwicklung des Servers fokussiert. Durch meine Tätigkeit als Werkstudent und aus persönlichen Interesse hatte ich Erfahrung mit dem Linux Terminal und Version Controlling und habe mich demnach auch in diesen beiden Themengebieten engagiert. Auch wenn die Aufgabenbereiche zu Beginn relativ klar abgegrenzt wurden kam es sehr schnell zu Überschneidungen und fast jeder Arbeitete in fast jedem Bereich mit. Ich wollte mich Beispielsweise eher auf das Back-End konzentrieren habe aber dann doch viel im Front-End gearbeitet da ich die im Back-End verarbeiteten Informationen im Front-End darstellen wollte. Auf Server Seite habe ich viel mit Fabian und Kevin gearbeitet im Front-End mit Graciela und Jack.

## 5 Projektmanagement

Wir haben das Projekt mit Hilfe von Agilen Methodiken entwickelt. SCRUM war hierbei das einflußreichste Vorbild in unserem Arbeitsprozess. Jeden Montag haben wir uns zusammen über Fortschritte und Hürden der letzten Woche ausgetauscht und über neue Ziele die in Zukunft erreicht werden sollen. Dies stellte unsere Sprints dar. Unser betreuender Professor - Prof. Dr.-Ing. David Stripppgen - kann Teilweise als Product Owner angesehen werden und Kevin Wrede als Scrum Master da er sich für die Einhaltung der Scrum Meetings eingesetzt hat. Unser Product Backlog wurde wie im Kapitel "Planung" erwähnt öfters angepasst und redefiniert. Dieser abgespeckten Form des SCRUM Vorgehensmodells fehlen einige Regeln und es gab zum Beispiel auch keine wirklichen Daily Stand up's.



Figure 2: LED Leiste

## 6 Hardware setup

Hardware war kein Schwerpunkt meiner Rolle im Team. Bei dem konzipieren und montieren des Smart Mirrors habe ich geholfen. Da wir den Smart-Mirror zu fünf zusammengebaut haben würde ich meinen Anteil mit 20% beziffern. Im späteren Verlauf des Projekts habe ich zusammen mit Kevin und Fabian den "Makey Makey" Game Controller in das System integriert um den Smart-Mirror mit hilfe des Makey Makey zu steuern. Bei der hardwaretechnischen Umsetzung habe ich verschwindend wenig beigetragen aber auf softwareseite konnte ich maßgebend mitwirken. Etwa ein drittel der technischen Umsetzung des Game Controllers geht auf mich zurück. Die optische Gestaltung hat Hjörðes Frese beigesteuert. Der Programmcode für den Makey Makey welcher durch die Webseiten Elemente iteriert beginnt in Zeile 513 der Datei "views/index.ejs" und endet in Zeile 535. Es wird eine Utility (jQuery.tabbable) benutzt und wir haben uns auf die Dokumentation auf selbiger Seite der Quelle<sup>1</sup> berufen. Dieser Code weiß anderen KeyDown-Events die Funktion hinzu durch Elemente zu "Tabben". Da die Tabulator-taste nicht dem Makey Makey zugewiesen kann und es keine "Shift+Tab"-Taste gibt mussten wir diesen kleinen Workaround anwenden. Kevin hat sich ein Sensor Kit für den RaspberryPi gekauft mit welchen wir experimentiert haben. Dazu gehören LED's, Feuchtigkeitssensor, Bewegungssensoren, Buzzer, Knöpfe, Geräuschsensoren und weitere Module die wir mit zusammen verkabelten und durch modifizierte Python Scripte kontrollierten. Wir haben alle für uns interessanten Sensoren zum laufen gebracht aber schlussendlich keine der Sensoren außer dem Funk Sender in das finale Projekt einbezogen. Dies lag daran dass die Sensoren sehr ungenau waren und wir schon verlässlichere Sensoren im System hatten oder wir keine Funktion für die Module in unserem System gefunden haben. Beim experimentieren mit den Sensoren schätze ich meinen Beitrag auf 50%.



Figure 3: Smart-Mirror

---

<sup>1</sup><https://github.com/marklagendijk/jquery.tabbable>

## 7 Server Setup

Mit dem aufsetzen des Servers war das Team und ich zu Beginn sehr lange beschäftigt. Wir haben mehrere Betriebssysteme ausprobiert und haben uns am Ende für Ubuntu-Core entschieden weil es nicht so überladen ist wie andere Distributionen und für Ubuntu auch sehr viel Dokumentation im Internet verfügbar ist. Ich hatte anfangs den NanoPi mehrmals mit hilfe einer micro SD geflashed aber konnte leider keine Serielle Verbindung herstellen da die Serielle USB Schnittstelle die ich benutzt habe defekt war. Wir hatten anfangs mehrere NanoPi's welche wir zusammen eingerichtet haben um mit verschiedenen Betriebssystemen zu experimentieren. Damit waren hauptsächlich Fabian, Kevin und ich mit beschäftigt. Nachdem wir alle NanoPi's geflashed haben habe ich mich zusammen mit den anderen um die grundlegende Einrichtung des Servers beschäftigt. Ich habe Node JS, NPM und EJS eingerichtet und GIT installiert. Desweiteren habe ich systemd so konfiguriert dass der Server automatisch beim hochfahren des NanoPi's gestartet wird. Beim einrichten des automatischen startens des Node JS service habe ich mich auf ein Tutorial im Netz<sup>2</sup> berufen. Kevin und Fabian haben sich eher um andere Punkte gekümmert beim einrichten des Servers. An und voran die Netzwerk konfiguration. Wir haben aber auch viele der aufgezählten Aufgaben zusammen abgearbeitet warum es mir schwer fällt eine Prozess tangabe meiner Leistung zu machen.

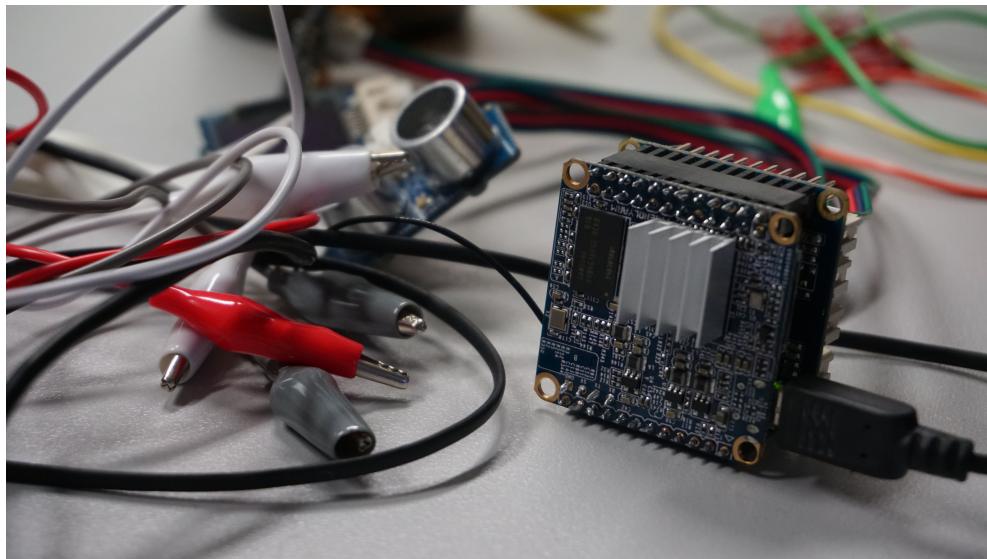


Figure 4: The NanoPi Neo AIR

---

<sup>2</sup><https://www.axllent.org/docs/view/nodejs-service-with-systemd/>

## 8 Softwareentwicklung

Der Löwenanteil meines Beitrags im Projekt bezieht sich auf die Entwicklung des Servers in Node JS und die Darstellung im Front-End über die JavaScript Template library EJS (Embedded JavaScript). Eine der ersten Meilensteine der Server Entwicklung war die Ausgabe eines Feuchtigkeitssensor im Front-End mit hilfe eines ESP8266. Um die direkte Kommunikation des ESP's dem Sensor und das senden der Daten an den Server hat sich allein Hjördes Frese gekümmert und sie hat mir auch beispielcode zur Verfügung gestellt welcher die Sensordaten im Server über Node JS in Empfang nimmt und als Variable speichert. Die Darstellung der Variable auf einer Tes-Weboberfläche habe ich nun eingerechnet und so hatten wir relativ schnell ein Grundgerüst auf dem wir aufbauen konnten. Dieser erster Erfolg geht weitestgehend auf Hjördes zurück und mein Anteil mit ca 20% am ursprünglichen code ist eher gering. Im Laufe des Projektes hat sich der Code auf Serverseite natürlich weiterentwickelt aber in Zeile 269 - 290 des server.js ist Code für die Verarbeitung der Sensordaten des Feuchtigkeitssensores in seiner letzten Form. Die Variable wird im index.ejs zum Beispiel in Zeile 90 benutzt. Im späteren Verlauf haben wir (Kevin, Fabian, ich) die Variablen der Sensoren alle in Objekte gespeichert um ein Objektorientiertes Design des Codes zu schaffen. Wir lernten hierbei viel über die Eigenheiten von JavaScript und Node JS. Nach den ersten Erfolgen hatten wir ein grundlegendes Verständnis von Get-Requests um Responses um mit den ESP's und anderen Geräten zu kommunizieren. Die Verarbeitung von Sensordaten war nun ein leichteres da wir das Konzept auf mehrere Module anwenden konnten. Ziemlich die gesamte server.js Datei ist eine aus ein Produkt aus Pair Programming mit Kevin und Fabian. Viele Änderungen und Erweiterungen sind aus gemeinsamen Programmieren entstanden und deswegen schätze ich den Betrag von uns drei auf jeweils 30% und 10% von Hjördes. Die Datei hat viele Änderungen gesehen aber im Vergleich zum Arbeitsaufwand ist sie relativ kurz mit nur 370 Zeilen. Eines der Aufwändigsten Segmente ist von Zeile 100 bis 189 die Verarbeitung der Informationen der WLAN fähigen Glühbirnen YeeLight von Xiaomi. Der Programmcode führt diverse Python Skripte aus die direkt mit den Glühbirnen kommunizieren können. Kevin hat diese Skripte mit Fabian erstellt und mit Hilfe des PythonShell Packets von Node JS ausführbar gemacht. Ich habe mich mit Fabian um die Verarbeitung in der server.js Datei gekümmert. Die Funktion "DiscoverBulbs()" wird alle 30 Sekunden ausgeführt und überprüft welche Glühbirnen im Netzwerk verfügbar sind und was für Status diese haben. Dazu gehört die Helligkeit, Name, IP, Farbe, und ob die Birne gerade angeschalten ist. Leider waren wir mit dem NanoPi AIR ressourcentechnisch sehr begrenzt und konnten die Statusabfragen nicht öfter durchführen. Damit die Glühbirnen immer in der richtigen Reihenfolge gespeichert (und angezeigt) werden habe ich eine kleine Funktion geschrieben die die YeeLights nach IP-Adresse sortiert (Zeile 141 - 152). Im Front-End haben wir nun die Lampen visualisiert mit Namen und einem Glühbirnen Symbol welches

sich je nach Status ändert ob die Glühbirne im Moment ein- oder ausgeschaltet ist. Die Lampen werden als Objekte in einem Array und index.ejs übergeben und in einem For-Loop werden auch so viele Lampen dargestellt wie es Glühbirnen im Netzwerk findet. Genau so funktionieren auch die Fenster und Pflanzen Sensoren. Die zugehörigen HTML Elemente werden ebenfalls dynamisch generiert. Die Glühbirnen ändern auch ihr Icon wenn diese angeklickt werden. Wenn das click event ausgelöst wird, wird die Funktion "changeIcon(id)" aufgerufen welche das Icon nach einer halben Sekunde austauscht. Die Funktion befindet sich in Zeile 412 - 436 im index.ejs. An der index.ejs Datei waren fast alle beteiligt. Graciela und Jack haben die statische Seite entworfen und in HTML/CSS geschrieben. Auch haben die beiden einige JavaScript Funktionen geschrieben welche zum Beispiel die Uhrzeit und Zeitungsartikel anzeigen. Die restliche Datei haben Kevin, Fabian und ich geschrieben wobei mein Fokus auf der dynamischen Generierung der HTML Elemente lag. Sehr zeitintensiv war auch das refreshen einzelner Inhalte der Webseite damit die Weboberfläche auch aktuelle Daten repräsentiert. Dazu mehr im Nächsten Kapitel.



Figure 5: Xiaomi YeeLight Glühbirne

## 9 Technische Hürden

Der Start in die Welt der IoT-Devices und Home Automation war ein wenig steinig da keiner von uns bis auf Kevin Erfahrung mit Smart Home Technologien hatte. Auch war hatten wir keinerlei Kenntnisse des Node JS Frameworks. Das flashen der Geräte hatte gleich zu Anfangs Probleme gegeben da einige Betriebssysteme nicht mit unseren Ein-Chip-Systemen kompatibel waren und einige unserer Hardware Geräte defekt waren. Nachdem wir die fehlerhaften Devices aber identifiziert hatten und kompatible Betriebssysteme gefunden hatten konnten wir aber recht schnell mit dem Eigentlichen Projekt beginnen. Die Arbeit mit Node JS schien mir auf den ersten Blick sehr verständlich und unkompliziert. Das aufsetzen eines Servers schien mit ein paar wenigen Zeilen Codes schon erledigt zu sein. Die erste ernüchternde Einsicht war dass wir sehr bald merkten das die Geräte öfters eine langsame Reaktionszeit haben und Statusabfragen manchmal bis zu einer Minute brauchen bis diese aktuell sind. Bei den YeeLights waren Wartezeiten als oftmals länger als eine Minute nötig. Dies hing auch damit zusammen dass die NanoPi Neo Air's Informationen deutlich langsamer verarbeiten können als ursprünglich erwartet. In unseren Testphasen und während mehrstündigen benutzens mussten wir auch verstetzen das der NanoPi sehr schnell überhitzt und komplett hängen bleibt. Glücklicherweise konnte Fabian einen PC Lüfter und einen Kühlblock an den NanoPi montieren damit er länger verlässlich ist. Unerwartete Probleme kamen auf als im späteren Verlauf das Server Team und gleichzeitig das GUI Team an den gleichen Dateien arbeiteten. Das GUI Team hat eine elegante Lösung um die Dateigrößen der vielen Icons zu reduzieren. Durch die Funktion "convertImgToSvg()" werden alle SVG Bilddateien in IMG-Tags zu inline SVG umgewandelt. Dies hat auch den Vorteil das die Bilder durch Programmcode manipuliert werden können damit beispielsweise die Farben der Icons auf der Mobilen Seite sich von den der Desktop Seite differenzieren ohne dass ein zweites Bild gebraucht wird. Der Code der von mir und dem Server Team geschrieben wurde ersetzt allerdings IMG-Tags wenn sich die Icons ändern sollen. Dies hatte zur Folge das sich die Icons plötzlich bei Click Events oder Aktualisierungen nicht mehr veränderten. Es hat eine Weile gedauert bis wir den Ursprung des Problems identifiziert haben und eine Lösung war nicht sofort parat. Wir haben es zunächst versucht die inline SVG's zu löschen und durch das richtige IMG auszutauschen welches wiederum sofort in inline SVG Code umgewandelt wurde. Dies hatte auf unseren Laptops den Anschein als würde es sehr gut funktionieren aber leider gab es merkliches Flimmern als wir den Code auf dem NanoPi Neo Air austesten. Die Leistung des NanoPi ist schlicht zu gering. Wir haben auch versucht direkt die Bilder als inline SVG zu manipulieren aber auch damit hatten wir Probleme. Nach mehreren Stunden haben wir uns dazu entschlossen beide Bilder beim aufrufen der Seite zu laden und konvertieren, für den Status "An" und "Aus". Nun haben wir bei einem Click Event den CSS Display Style auf "none" oder "inline" für beide Bilder

gesetzt damit immer nur ein Bild sichtbar ist. Dies war zwar leider nicht die Eleganteste Lösung aber auf jeden Fall die schnellste. Das automatische ändern der Icons bei Statusänderungen war ebenfalls eine große Herausforderungen mit vielen Hindernissen. Anfangs hatten wir nur einen automatischen Page Refresh der in einem Intervall die gesamte Webseite neu geladen hat. Dies war keine längerfristige Lösung aus naheliegenden Gründen. Da sich viele Status ändern können wenn zum Beispiel von einem anderen Computer oder Handy ein Gerät ein/aus geschaltet wird, muss sich die Weboberfläche ständig aktualisieren um korrekte Informationen anzuzeigen. Wir haben deswegen zunächst eine Mechanik eingebaut die einen Request auf die Seite macht um die Variablen(Objekte) zu aktualisieren und dann mit den Status der Webseite abgleicht und im Falle einer Abweichung das Bild wechselt. Bedauerlicherweise hat die Funktion nicht immer korrekte Ergebnisse geliefert. Es hat oft zu lange gedauert bis die richtigen Ergebnisse angekommen sind und ab und zu hat sich der Status verändert ohne dass eine Änderung im System vom Nutzer vorgenommen wurde. In der Funktion "refreshImages()" haben wir Tagelang experimentiert und wir haben dort noch Codeschnipsel in Kommentare geschrieben die wir bisher benutzt haben. In der letzten Version unseres Systems ruft die Funktion nur einen Request auf die Seite auf um die Variablen zu aktualisieren und lädt dann mit Hilfe von jQuery die DIV's neu welche Inhalte umschließen die sich ändern könnten. Das benötigte jQuery Wissen haben wir uns auf w3schools.com/JQuery/ angeeignet. Der Code hierzu ist in der index.js Datei in Zeile 361 - 407. Eine kleine Eigenheit von EJS ist wie ich raus finden musste dass wenn ich über einen Array von Objekten mit einem For-Loop iterieren will dass das erste Objekt existieren muss da auch der Code innerhalb des Loops ausgewertet wird für den ersten Index obwohl der Array leer ist. Wenn der Array leer ist aber man innerhalb des For-Loops auf eine Variable der Objekte im Array zugreifen will bekommt man eine Nullpointerexception. Dies habe ich mit einem Workaround gelöst indem ich immer mindestens ein Objekt übergeben habe mit bestimmten Properties welche ich später identifizieren kann das Objekt im Nachhinein zu löschen. Das wollte ich vermeiden aber ich konnte keine andere Lösung finden. Es gab weitere Probleme bei deren Lösungsfindung ich beteiligt war aber schlussendlich nicht viel beitragen konnte. Dazu gehört zum Beispiel das finden und Konfigurieren eines Browsers der auf dem Raspberry Pi Zero läuft und gleich beim booten im Kiosk Mode startet und das benutzen von "Der Zeit"-API.



Figure 6: Programmier Session

## 10 Showtime

Vor der Showtime haben wir unsere gesamte integrierte Hardware Infrastruktur miteinander verbunden und alle Sensoren und Geräte ausgiebig getestet. Es gab einige kleinere Fehler die bis dato noch nicht aufgefallen sind wie zum Beispiel Probleme beim Layout und Sensornamen die nicht richtig angezeigt wurden. Sonst hat das System erwartungsgerecht funktioniert und wir konnten uns bald der Vorbereitung für die Showtime widmen. Ich selbst war weniger an der Gestaltung und Präsentation beteiligt sondern widmete mich mehr einigen letzten fixes des Servers und ich wollte noch mit Graciela Buttons einbauen welche die Farben der YeeLights steuern. Der Code hierzu ist im Grunde fertig geworden aber wir hatten noch ein paar Probleme bei der Darstellung der Farboptionen und haben uns somit dazu entschieden die Funktionalität wieder raus zu nehmen. Der Code befindet sich immer noch auskommentiert im Projekt. Am Tag der Showtime hatten wir leider einige technische Probleme welche wir darauf zurückführen dass das Netzwerk überlastet war mit so vielen Projekten und Geräten auf kleinen Raum.

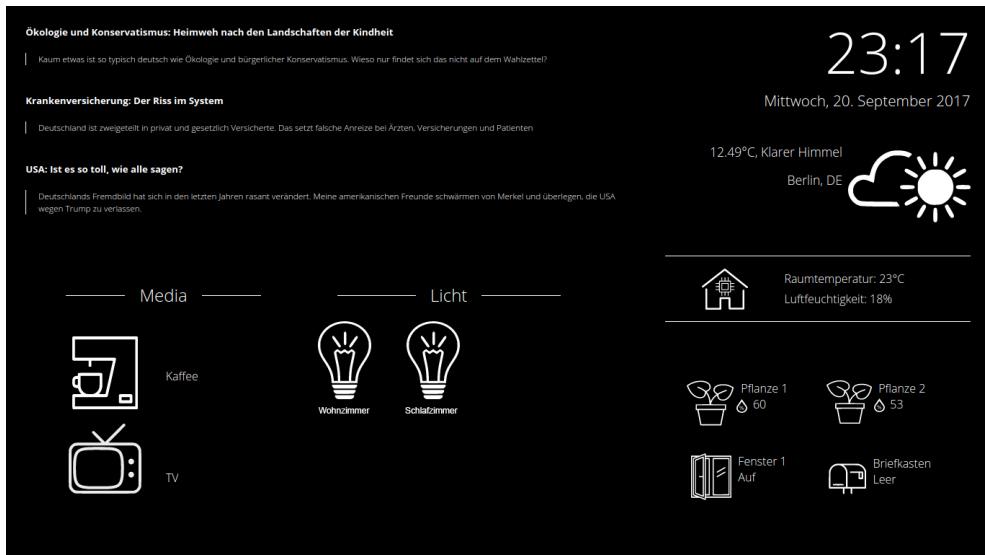


Figure 7: Finale Weboberfläche

## 11 Fazit und Ausblick

Rückblickend kann ich nur betonen dass mir das Projekt viel Spaß gemacht hat und ich sehr zufrieden war mit der Thematik des Projekts und meinem Team. Ich hatte schon im Vorfeld Interesse an Home Automation und dies war die Perfekte Gelegenheit mich mit dem Thema tiefer zu befassen. Durch das Projekt konnte ich meine Programmierkenntnisse ausbauen und habe viel über Server Architekturen gelernt sowie über die Funktionsweise von Sensoren und Mikrocontroller. Ich hatte den Eindruck dass jeder einzelne im Team sehr motiviert war und wichtige Beiträge zum Projekt beisteuern konnte. Wir haben uns sehr Regelmäßig getroffen und wir konnten uns gegenseitig sehr gut helfen. Unser Projektleiter hat uns viel Freiraum gelassen um Ideen zu verwirklichen welche wir für interessant hielten und war auch immer für Fragen erreichbar und konnte uns sehr gute Tipps geben.

Da ich das System auch Zuhause nutzen werde, plane ich einige Details zu erweitern bzw. anzupassen. Die Funktion für das ändern der Farbe der YeeLight Glühbirnen werde ich hinzufügen und vielleicht die Möglichkeit Geräte zu benennen und die Webpage anzupassen nach eigenen Bedürfnissen. Da die Arbeit am Projekt mit dem Team so gut geklappt hat finde ich es schade dass wir unseren Fokus wieder auf andere Vorlesungen und Seminare legen müssen. An Ideen für Features die wir in Zukunft einbauen könnten fehlt es auf keinen Fall. Das System könnte Module integrieren die der Sicherheit des Benutzers dienen. Vibrationssensoren die an Fenstern angebracht werden können um Einbrüche zu erkennen um den Nutzer warnen und/oder Überwachungskameras die Bilder/Videos an den Benutzer schicken sobald Bewegung im Haus aufgezeichnet wird. Der Einbezug einer Sprachsteuerung wäre ebenfalls eine interessante Idee. Die Regulierung der Heizkörper ist auch ein Feature das in einem Smart Home nicht fehlen dürfte. Ich hoffe das System wird sich noch in Zukunft weiter entwickeln und ich freue mich schon auf das Master Projekt.