RC LEGO CAR PROJECT

PHYSICAL WEB MEETS WEB BLUETOOTH

BLUETOOTH BACKGROUND

GENERAL INFORMATIONS

- developed 1990
- standard for wireless communication device to device communication
- current version: v4.2
- „normal“ Bluetooth connection oriented (pairing)
- Bluetooth Low Energy (since v4.0) ad hoc communication possible

BLUETOOTH PROFILES

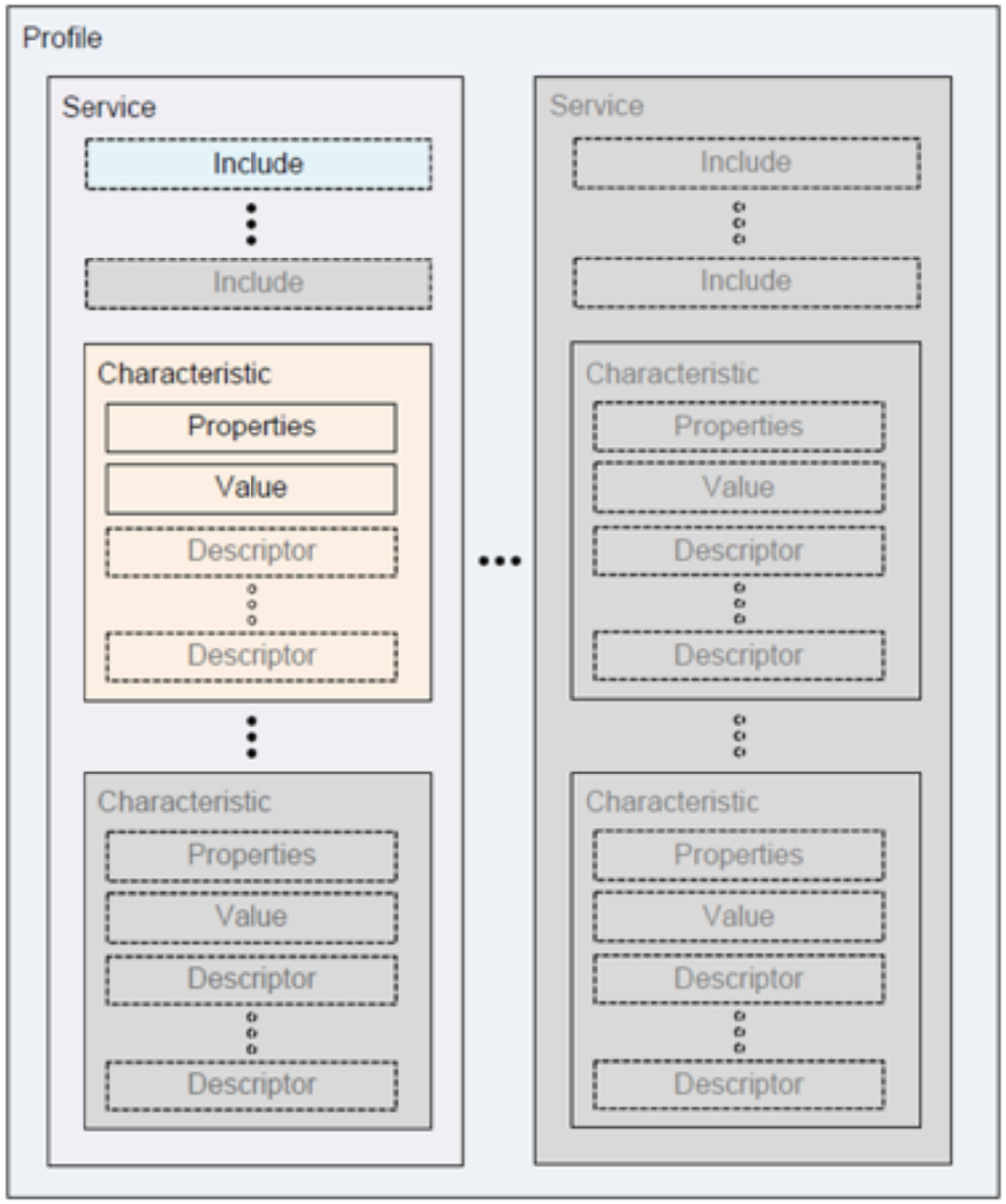
- Interface definitions. Samples are:
  - **A2DP** (Advanced Audio Distribution Profile) - audio streaming
  - **OBEX-FTP** - file transfer
  - **HID** - Human Interface Device Profile
  - **PAN** - Personal Area Network Profile
  - **GATT** - Generic Attribute Profile

BLUETOOTH LOW ENERGY

- Since Bluetooth v4.0
- Optional (v4.0 devices don’t have to support BLE)
- Physical wireless communication is different for Bluetooth and BLE so most Bluetooth chips have to circuits
- Slaves send advertisement packages periodically
- only GATT profiles supported
- Android: since Version 4.3, iOS: since Version 5

BLUETOOTH GATT PROFILE

- A device implementing the GATT profile provides:
  - 1..n services [UID]
    - mit 1..n characteristics [UID]
      - read/write/readWrite
      - Represents a value
- There are a number of standardized services (standardized UID):
  - Battery Service
  - Blood Preassure
  - ...



BLUETOOTH BEACONS

- Based on Bluetooth Low Energy
- Continuous advertisement package broadcasting
- Protocols
  - iBeacon (Apple): UID advertisement
  - AltBeacon: UID advertisement
  - URIBeacon: URL advertisement
  - Eddystone (Google): UID, URL or TLM advertisement
- Clients use the beacons to react to their presence and / or data

PHYSICAL WEB

- Initiated by Google
- Idea: Interaction with a nearby device should be possible without installing anything on the client device
- Based on Eddystone Beacons (currently, the idea is independent from that)
- Integration in Android devices is currently rolling out (newest Google Play Services -> Nearby)
- Usable with the Physical Web app (Android) or Google Chrome (iOS)

WEB BLUETOOTH

CORNERSTONES

- Standard for interacting with BLE devices from a web pave via JavaScript (WebApp)
- Developed as a community team managed from the W3C
- No standard yet and no plans in sight
- Many years will pass until WebBluetooth is a standard and integrated in all common browsers and platforms

CURRENT PROGRESS

- Google has done a first reference implementation in Chrome/ Chromium
  - Work in progress: API can change any time
    - Only a hand full of platforms supported (27.06.16: Android, Chrome OS, Linux, planned: Windows, MacOS, iOS: when Apple adds WebBluetooth to its WebKit)
- Microsoft is thinking about an implementation („under consideration“)
- Apple didn‘t release any statement regarding WebBluetooth yet

SECURITY

- Protection against abusing this API is the primary goal of the specification
- Requirement: User has to actively interact with the device to provide a web application access to a BLE device
- The web app can’t see all BLE devices. It only gets access to the device the user selected (so that location determination isn’t possible using this API)
- API only can be used having a secure connection (https)

RC LEGO CAR PROJECT

INTENTION

- Combine WebBluetooth with the ideas from the Physical Web
- Controlling a device
  - without installing an app
  - without having the device to be connected to the internet
  - using state of the art web technology like offline caching

WHY LEGO?

- First try: Switching a LED on and off
  - BLE Slave: Raspberry Pi (NodeJS)
- Not very impressive => the idea to control a Lego Technic car has been born
  - Project mutated from a web to a electronic project

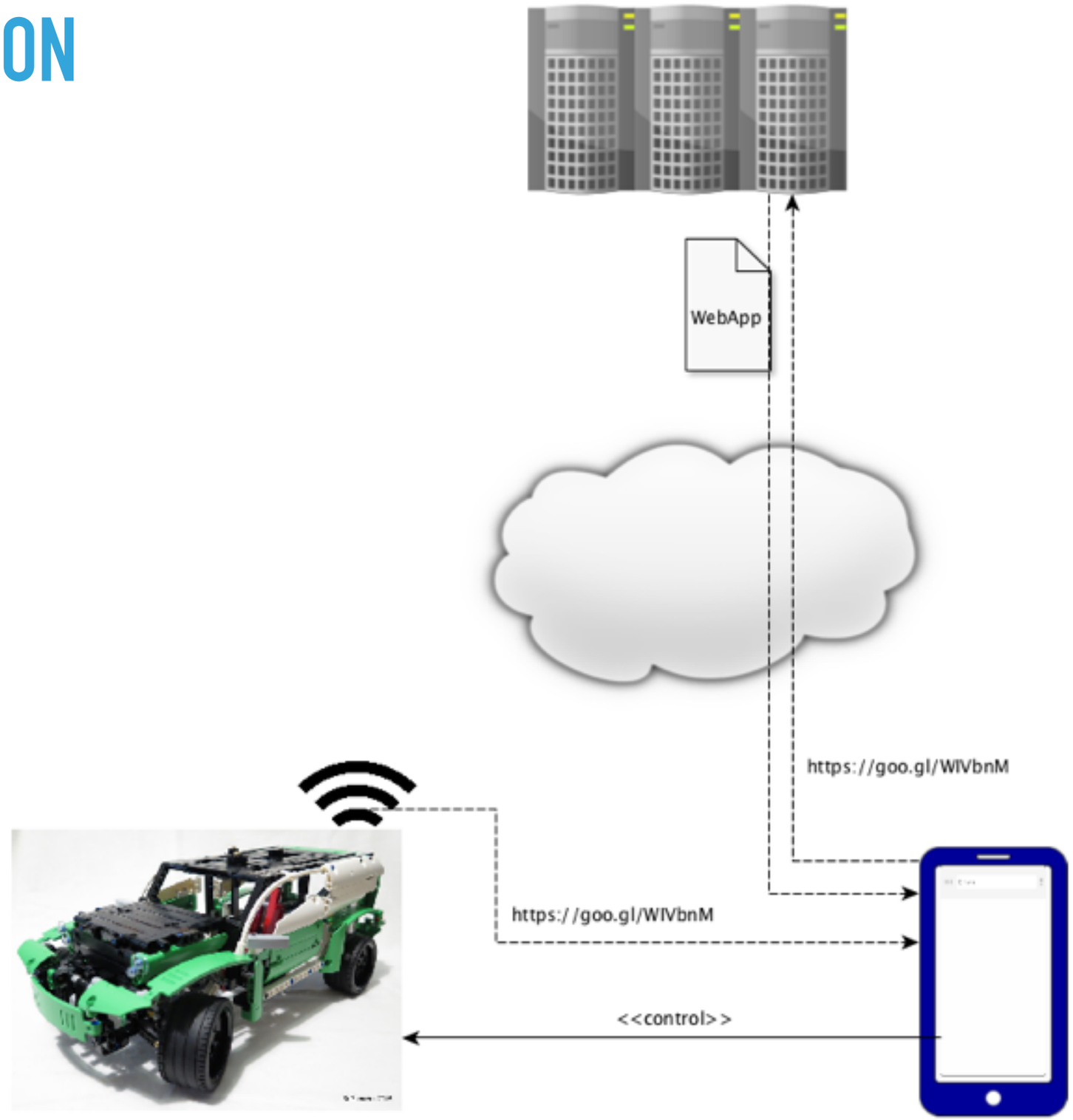
VARIANTS

- Over the course of the project different variants were implemented
- BLE Slave (Car logic):
  - Raspberry Pi (**NodeJS**)
  - Intel Edison (**NodeJS**)
  - RedBear Duo (**Arduino**) and ATMega32 (**C**)
- BLE Master (Remote control)
  - Android App (**Java, Android Framework**)
  - Android Wear App (**Java, Android Framework**)
  - Tizen Wear App (**HTML 5, TAU [Tizen Advanced UI]**)
  - Web App (**HTML 5, AngularJS 2**)

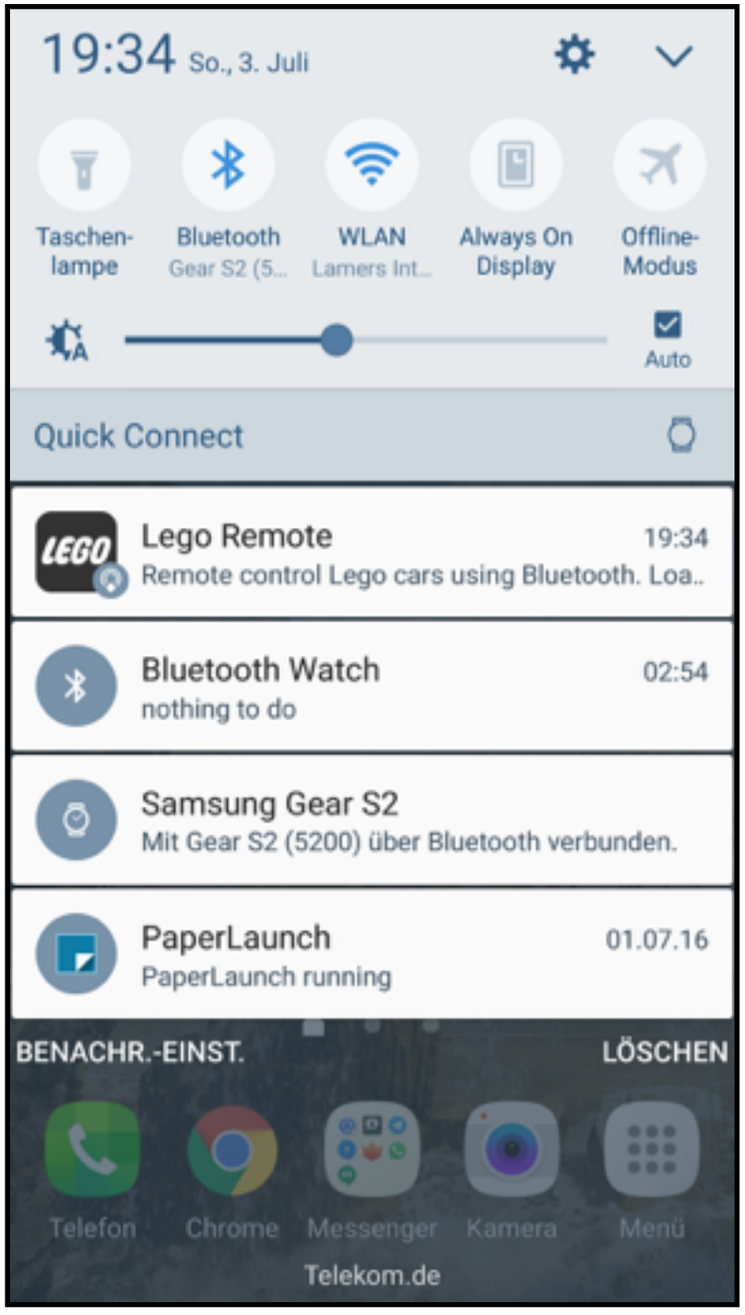
RC LEGO CAR PROJECT

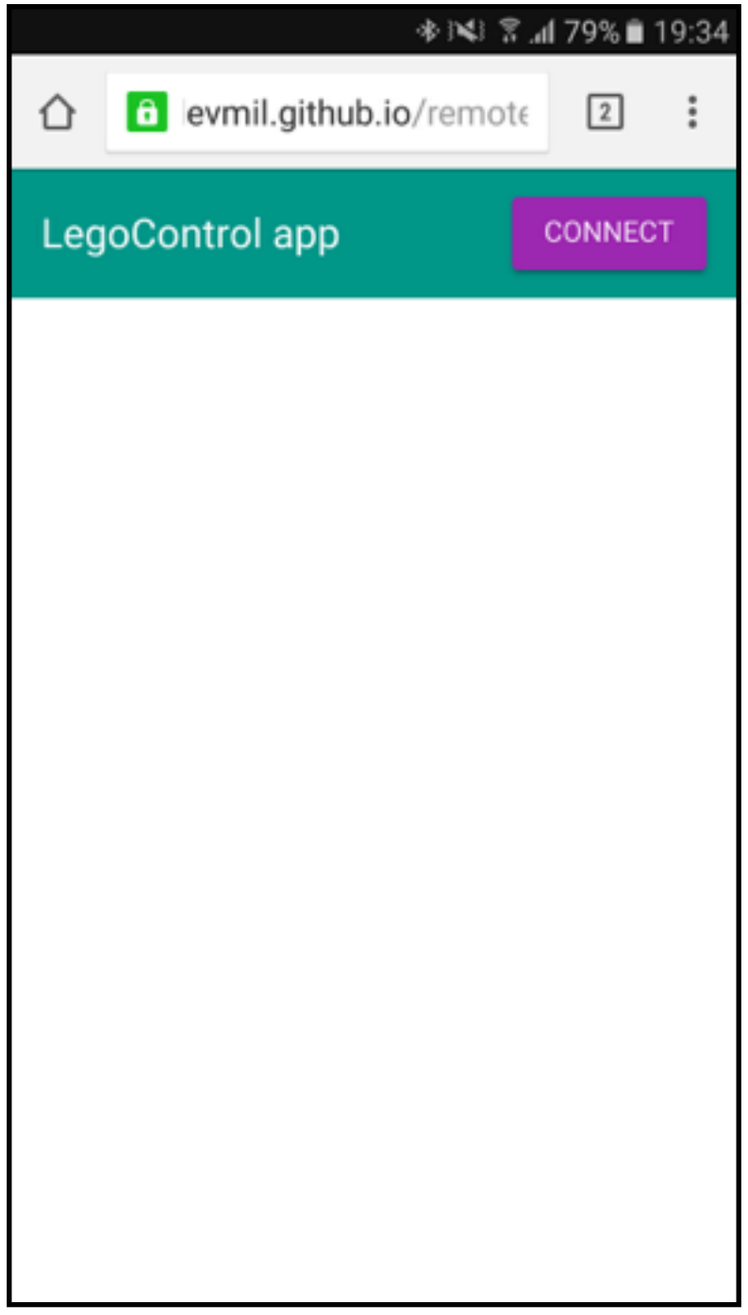
REMOTE CONTROL

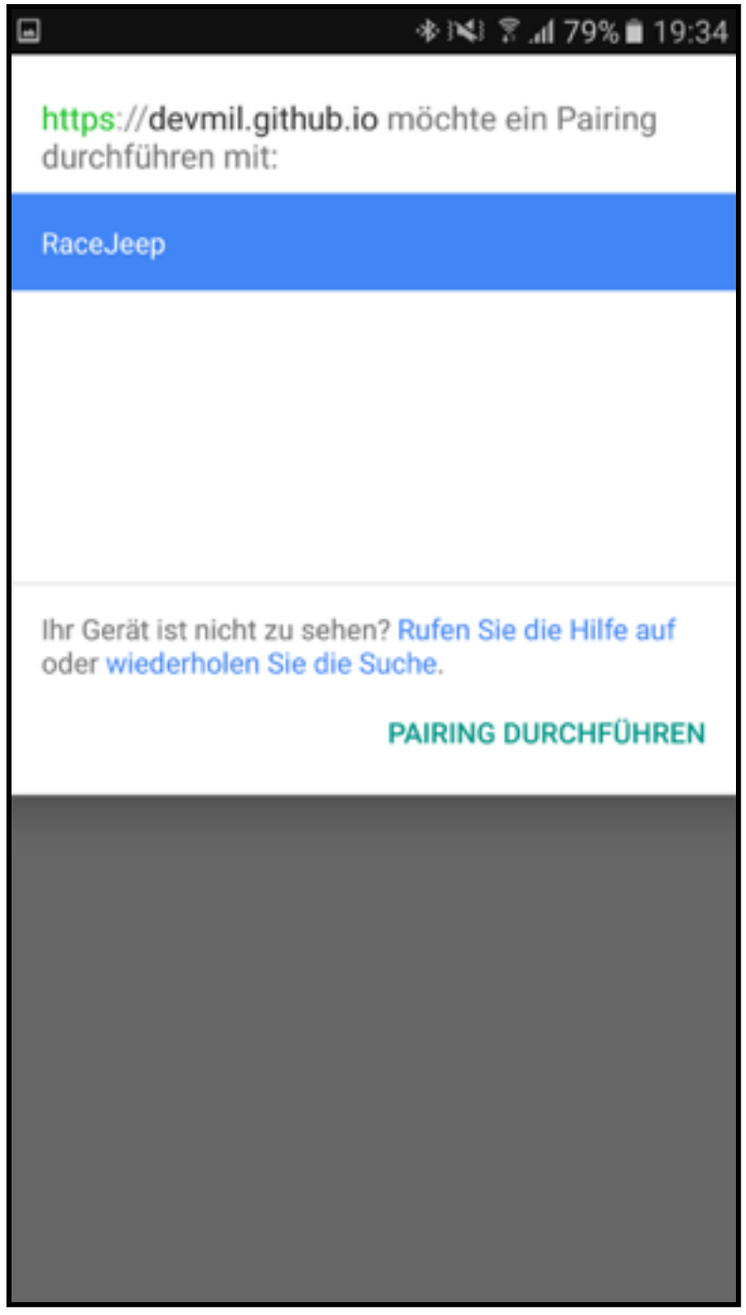
CONNECTION



SMARTPHONE UX

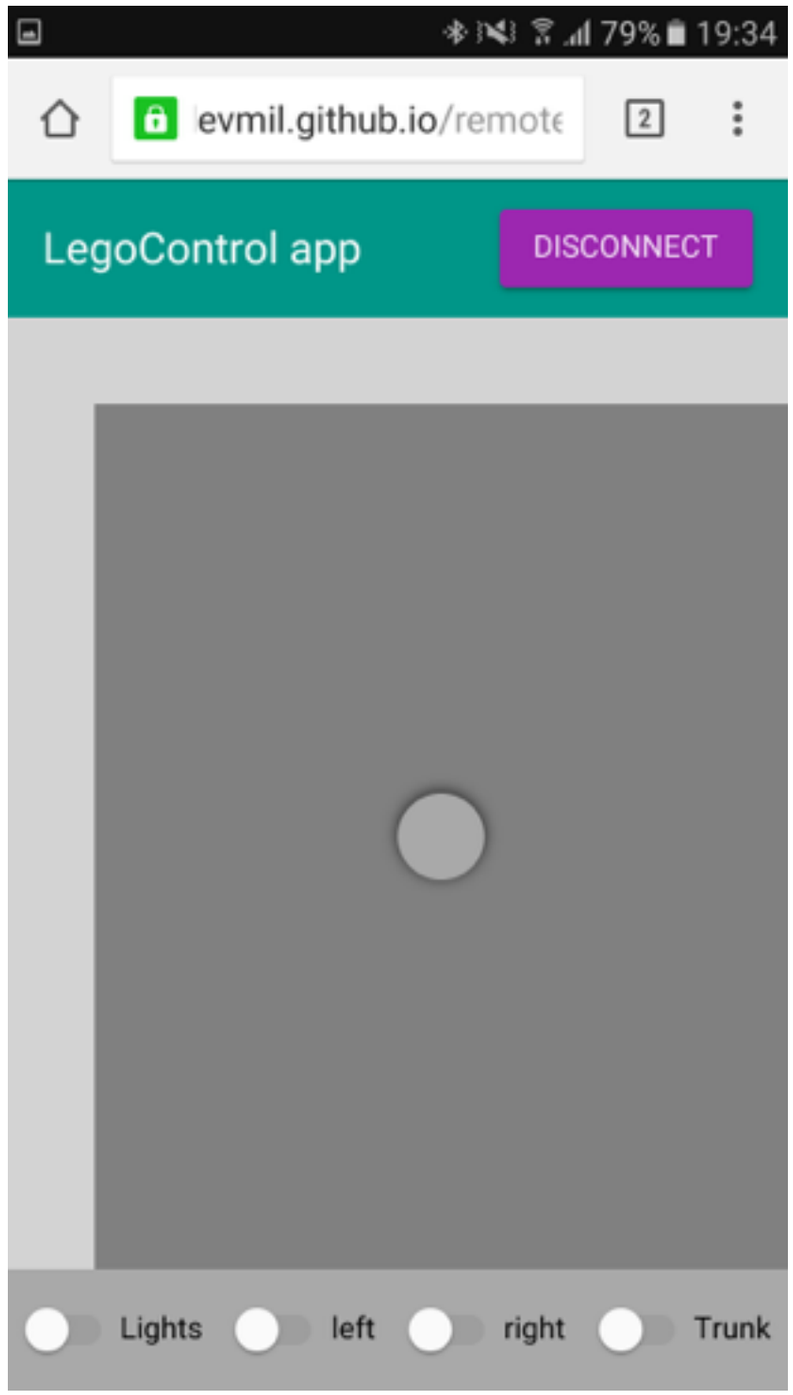


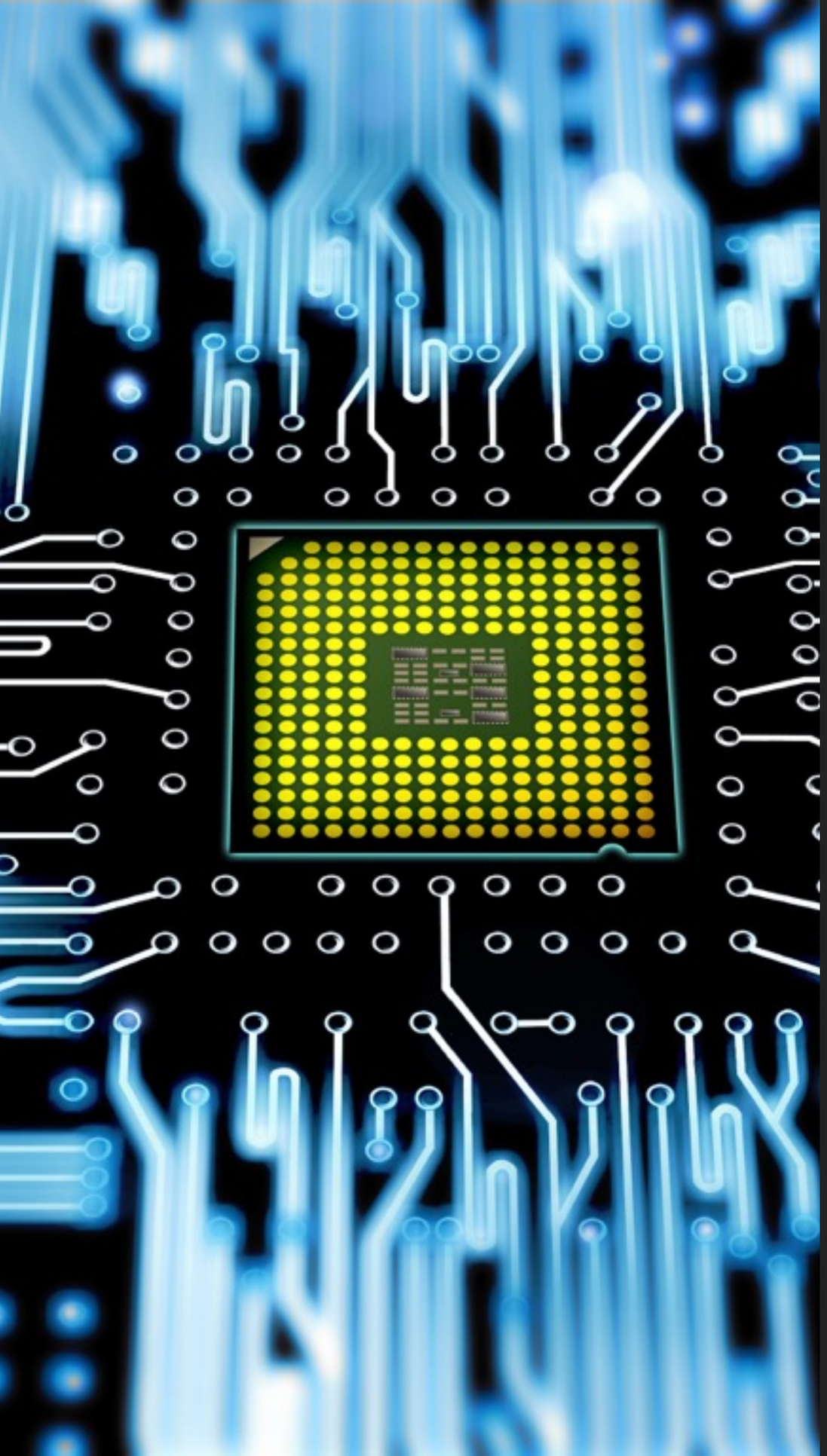




WEBAPP

- Typescript
- AngularJS 2
- Connects to the Lego control service (via UID)
- Writes data to characteristics:
  - SpeedCharacteristic
  - SteeringCharacteristic
  - TrunkCharacteristic
  - FrontLightCharacteristic
  - BlinkCharacteristic

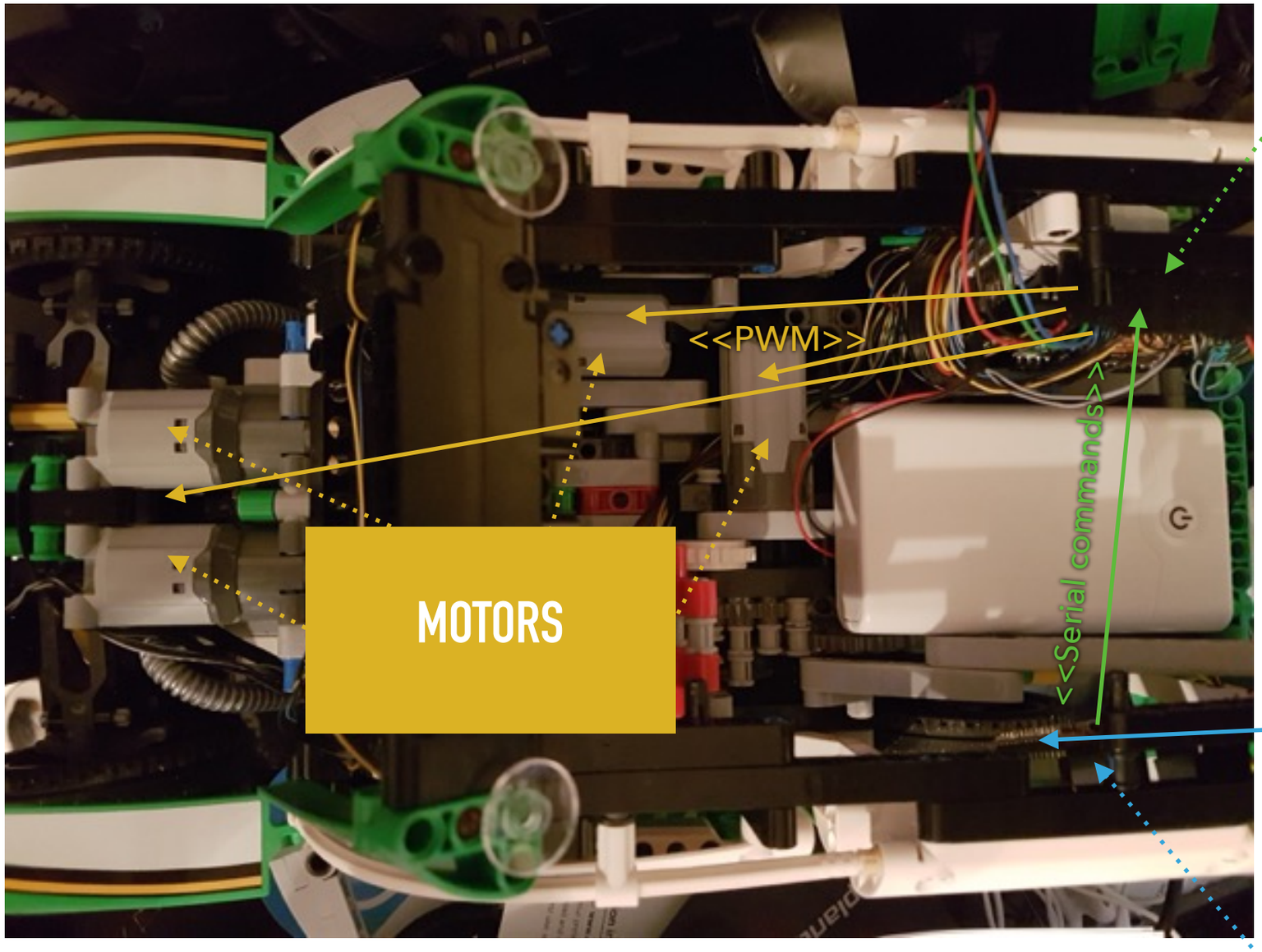




RC LEGO CAR PROJECT

CAR ELECTRONICS

ARCHITECTURE

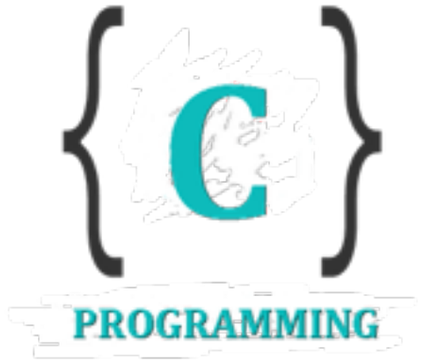


MAINBOARD

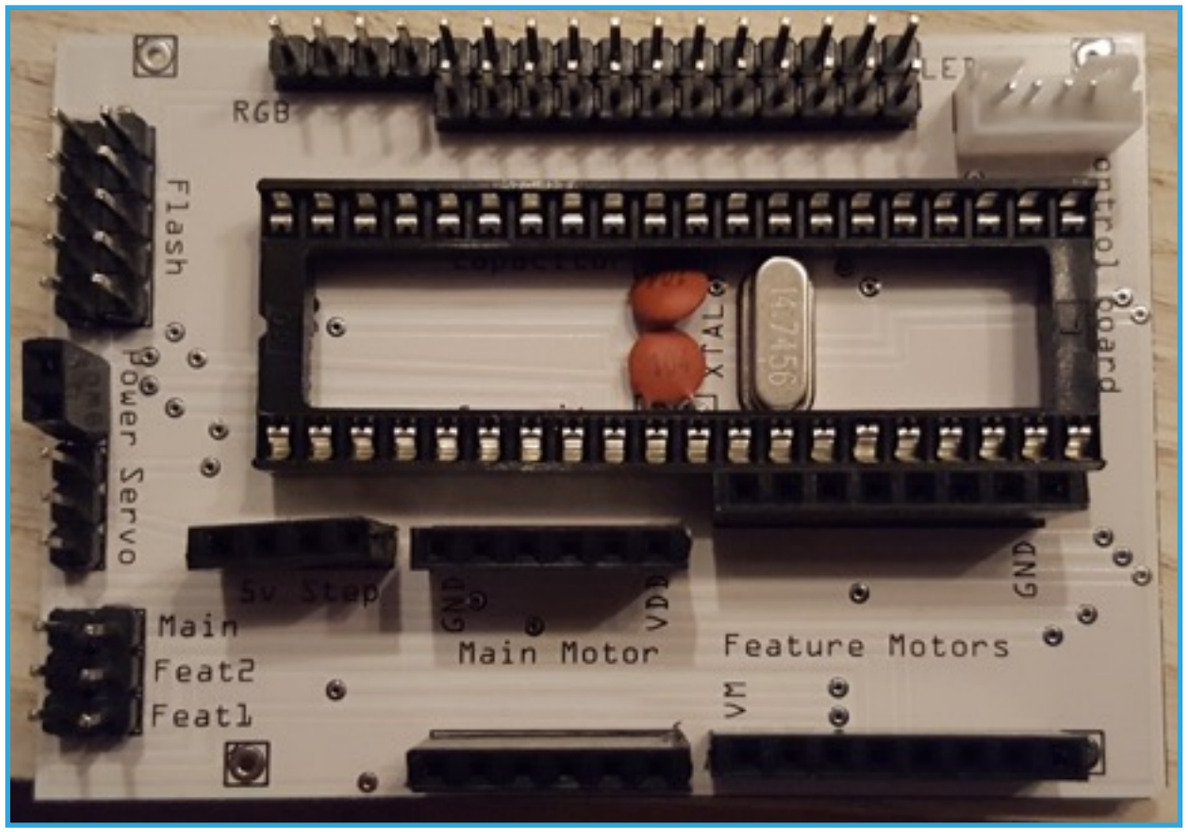
<<BT Characteristics>>

CONTROL BOARD

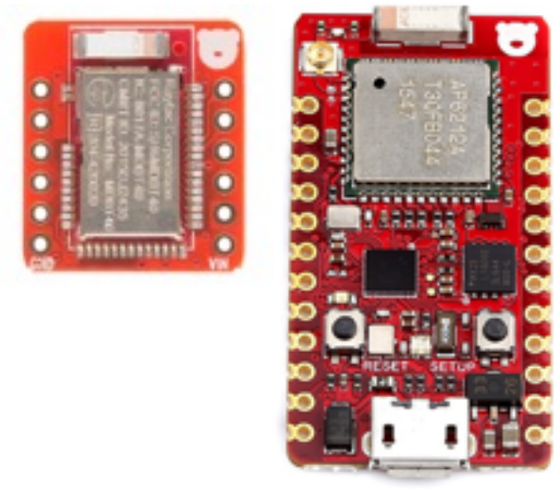
MAINBOARD



- Hardware control
- No car specific logic
- Timer for PWM signals
- Parts:
  - ATMega32
  - 5V Step down regulator
  - H bridges (single for main motor, dual for feature motors)
- Receives commands via UART from control board



CONTROL BOARD



- Provides the BT service
- Contains the car logic
  - what does „open trunk“ mean?
  - Blink control (sends LED on/off to the mainboard, controls the timings itself)
  - has a car specific profile
    - Maximum steering angles
    - is steering inverted?
    - is Trunk, Front light, Blinking, … supported?
- Broadcasts the Eddystone telegram that contains the URL to the WebApp

SOURCES

- Wikipedia
  - Bluetooth: https://de.wikipedia.org/wiki/Bluetooth
  - BLE: https://de.wikipedia.org/wiki/Bluetooth_Low_Energy
- Bluetooth developer
  - GATT: https://developer.bluetooth.org/TechnologyOverview/Pages/GATT.aspx
- Physical Web
  - https://google.github.io/physical-web/