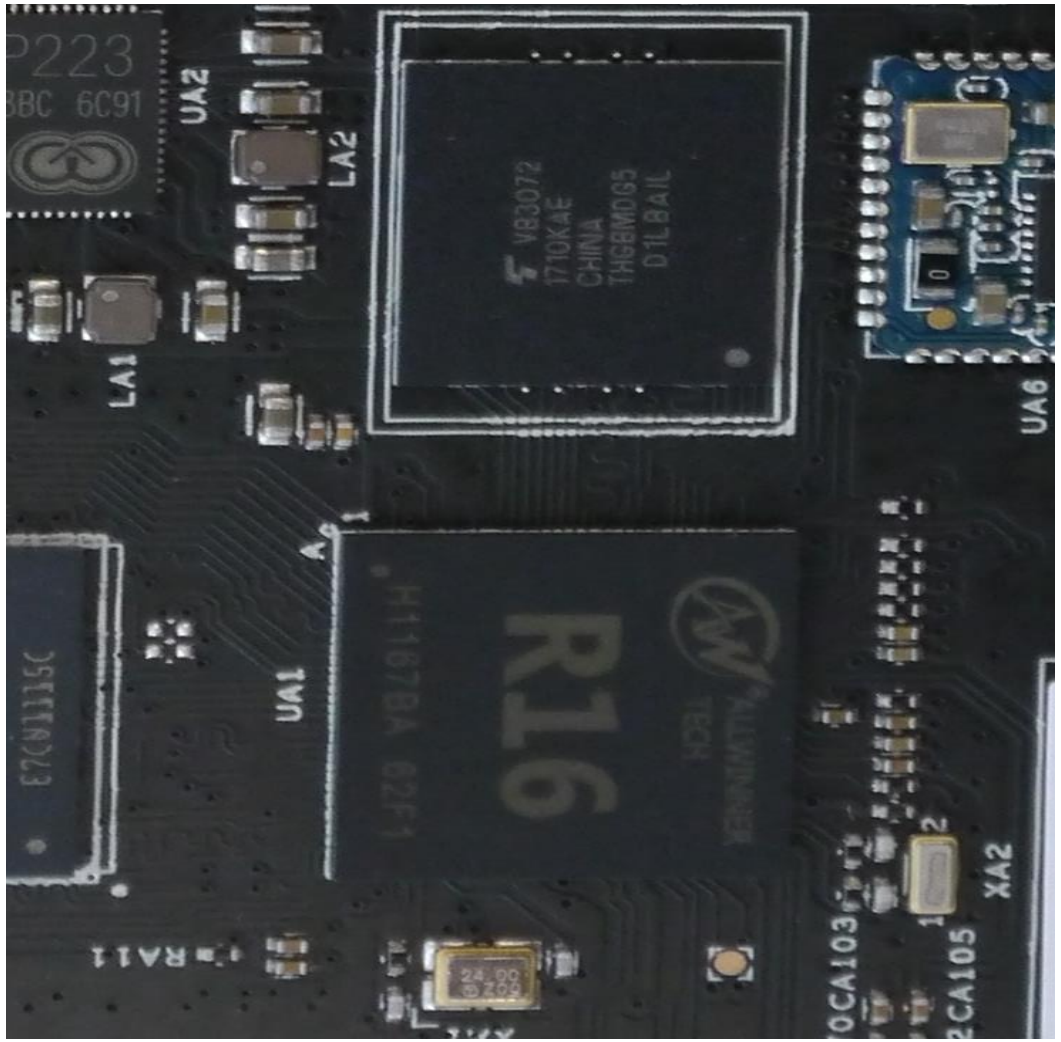
Unleash your Smarthome devices: Vacuum Cleaning Robot Hacking

Dennis Giese & Daniel Wegemer

Overview hardware

- Application-CPU: Allwinner R16 SoC (=A33)
  - Quad-Core ARM Cortex-A7 @ 1.5 GHz
  - RAM: 512MByte (various Chips)
  - FLASH: 4GByte (Toshiba THGBMDG5D1LBAIL)
- Sensor-CPU: STM32F103VET6
  - ARM Cortex-M3
- LIDAR-CPU: TI S320F28026DAS
- Power Chip: AXP223
- Battery: 14.4V, 5200mAh, ATT_SWD_LG
  - Charging control IC: CHRIC_BQ24773

Pin Layout CPU



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
					UART0							MMC2				MMC1		
A							MMC Reset	D6	D4	D2	D0		D2	D0	CLK		TX	UART1
B							D7	D5	D3	D1			D3	D1	CMD		RX	
C										CLK			SDA					TWI1
D							RX	TX		CMD			SCL					
E																		
F																Recovery	Confirm	UART2
G																RX	TX	
H															Line IN L			
J															LINE IN R			
K															PHONE IN			
L															PHONE IN			
M															PHONE		MIC1P	
N															PHONE		MIC2P	
P															SDA	SCK	RESET	RSB0
R																		
T							LCD9	LCD7	LCD5	LCD3	LCD1					USB-DM0	USB-DP0	USB 1
U							LCD8	LCD6	LCD4	LCD2	LCD0	USB DRV				USB-DM1	USB-DP1	USB 2
	DRAM		VCC/VDD		GND						LCD							

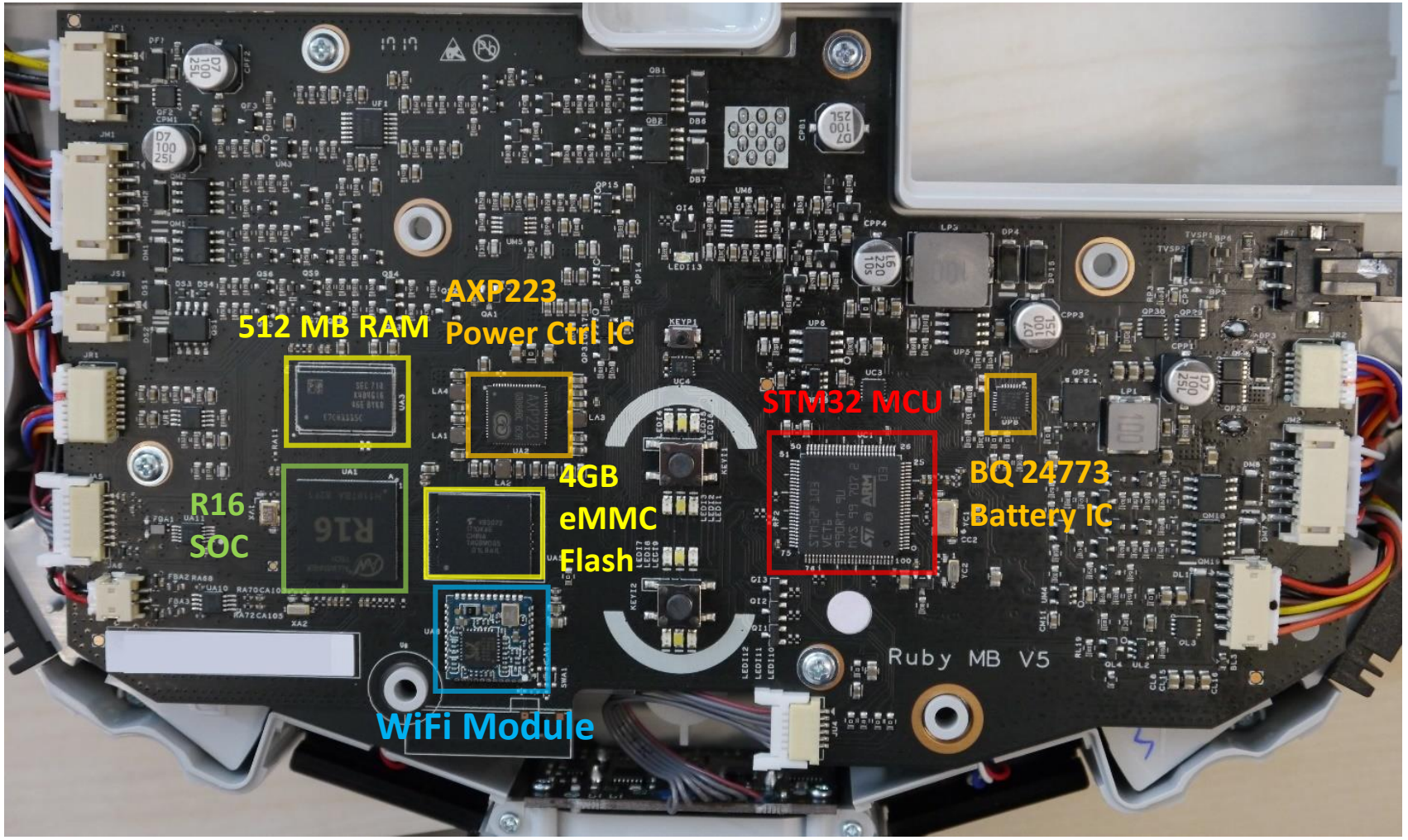
Overview sensors

- 2D LIDAR SLAM (5*360°/s)
- Ultrasonic distance sensor (front)
- 4x IR sensor (bottom side, cliff sensors) SHARP 0A51SK
- 1x IR sensor (right side, wall sensor) SHARP 0A51SK
- Senodia *ST480*
  - 3-axis Magnetic Sensor
- Bosch BMI160
  - digital tri-axial accelerometer + digital tri-axial gyroscope
- magnetic sensor for „wall“ feature (unknown)
- Nidec fan with speed sensor
- speedmeter for wheels
- dustbin sensor
- 2x bump sensors

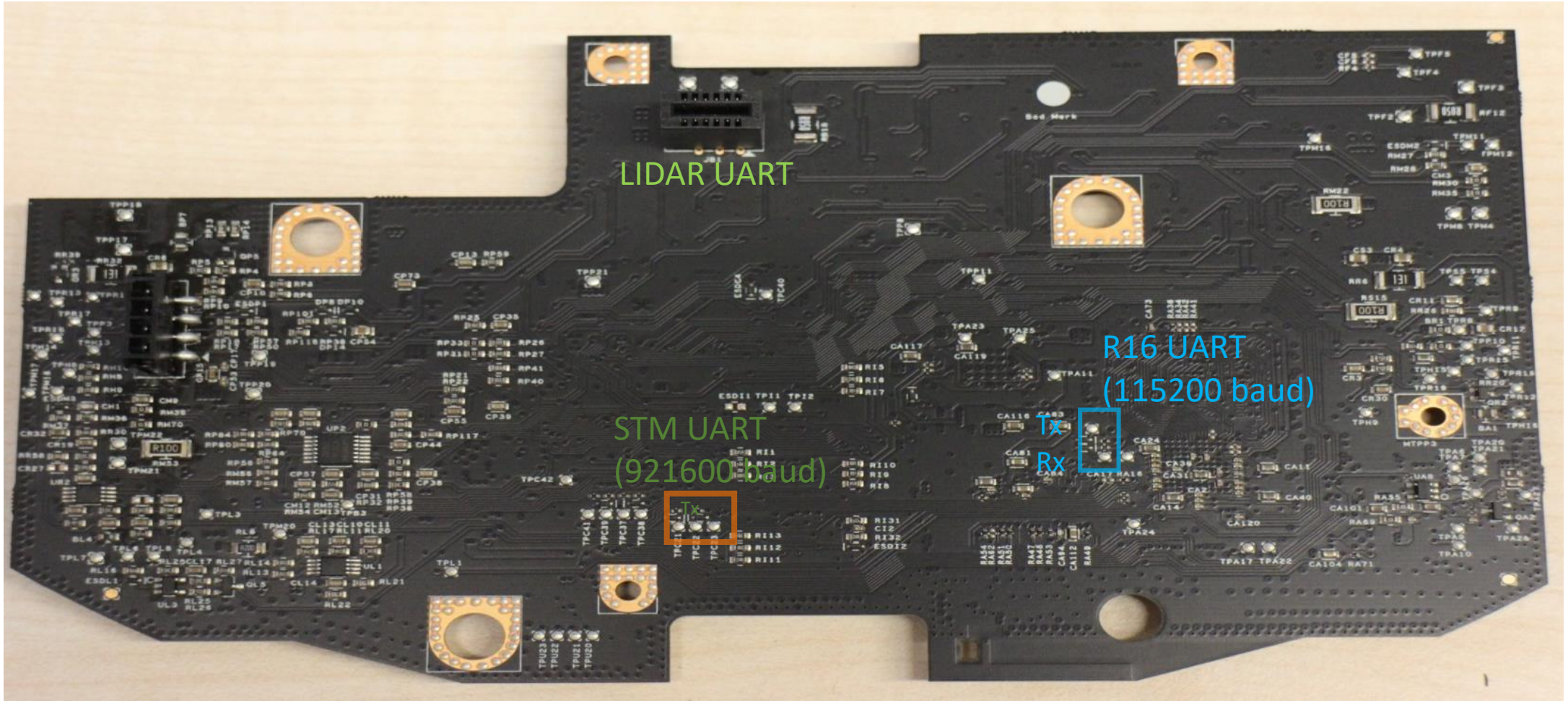
Overview connectivity

- Wifi: F89ETSM13-W2
  - Realtek RTL8189ETV 11n WIFI Module
  - IEEE 802.11 b/g/n 2.4GHz 1T1R SDIO
  - Connected via 4-Bit Mode SDIO
- USB 2.0
  - Host and Client mode
  - By default custom adbd
- UART
  - Communication between CPUs, LIDAR via UART
  - Accessable via testpoints
  - Dedicated UART for serial console of R16 SoC

Frontside layout mainboard



Backside layout mainboard



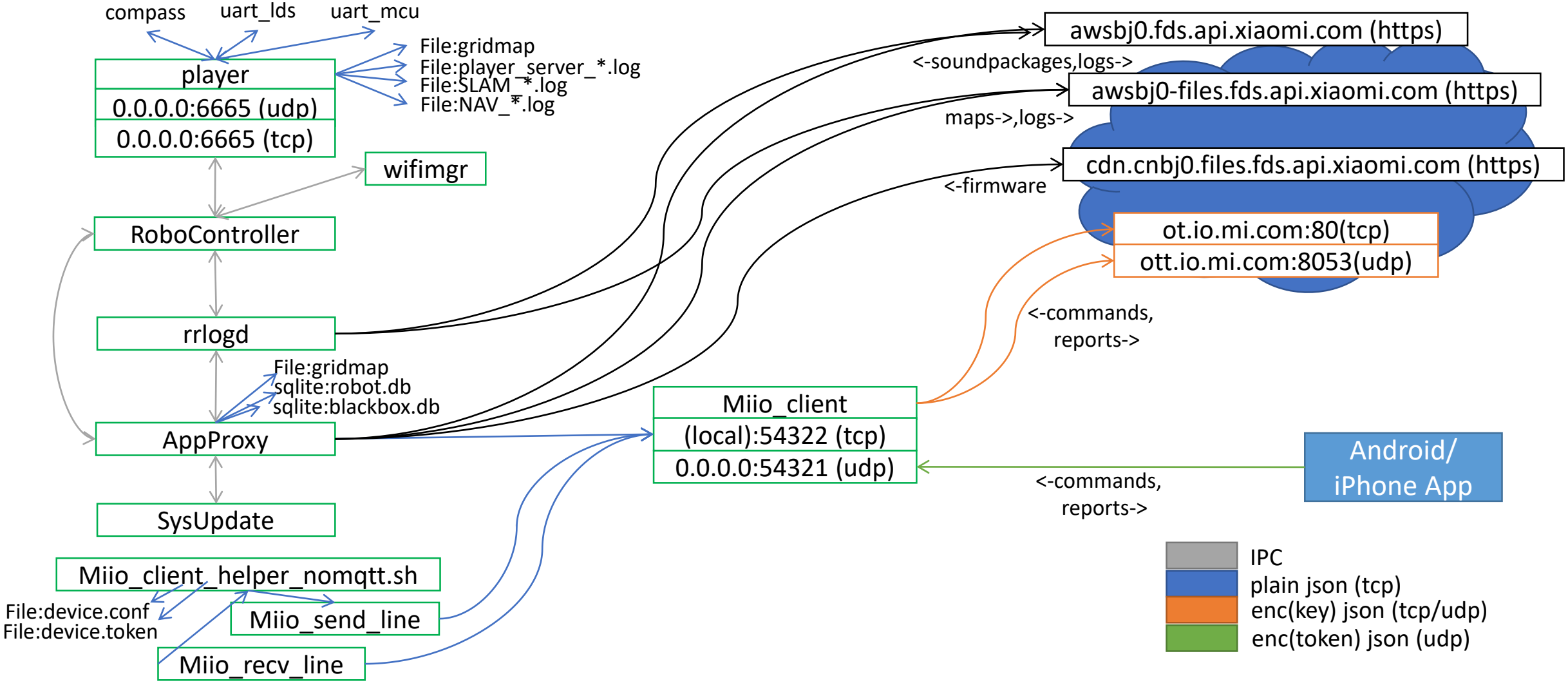
eMMC Layout

Label	Partion nand{}	Size in MByte	Start address
boot-res	a	8	0x00008000
env	b	16	0x0000c000
app	c	16	0x00014000
recovery	d	512	0x0001c000
system_a	e	512	0x0011c000
system_b	f	512	0x0021c000
Download	g	528	0x0031c000
reserve	h	16	0x00424000
UDISK	i	~1900	0x0042c000

eMMC Layout

Label	Content	Mountpoint
boot-res	bitmaps & some wav files	
env	uboot cmd line	
app	device.conf (DID, key, MAC), adb.conf, vinda	/mnt/default/
recovery	fallback copy of OS	
system_a	copy of OS (active by default)	/
system_b	copy of OS (passive by default)	
Download	temporary unpacked OS update	/mnt/Download
reserve	config + calibration files, blackbox.db	/mnt/reserve/
UDISK	logs, maps, pcap files	/mnt/data

Communication relations



Firmware updates

- Full and partial images
  - Encrypted tar.gz archives
  - Full image contains disk.img
    - 512 Mbyte ext4-filesystem
  - Partial image contains only content of /opt/rockrobo
- Encryption
  - Static password: “rockbobo”
  - Ccrypt [256-bit Rijndael encryption (AES)]
- Integrity
  - MD5 provided by cloud

Sound packages

- Contents of /mnt/data/sounds
  - Encrypted tar.gz archives
  - Contains wav-files in specific language or style
  - Default soundfiles are in /opt/rockrobo/ressources/{prc,tw}
- Encryption
  - Static password: “r0ckrobo#23456”
  - Ccrypt [256-bit Rijndael encryption (AES)]
- Integrity
  - MD5 provided by cloud

Update process

1. App sends encrypted packet with pkg info
   - *milO.ota {"mode":"normal“, "install":"1", "app_url":"https://[URL]/v11_[version].pkg", "file_md5":“[md5]",”proc":"dnld install“}*
2. Device downloads pkg
3. Device verifies MD5 and decrypts pkg to „Data“
4. Unpack pkg and ‚dd‘ to partition „Download”
5. ‚dd‘ to partition „system b“ from „Download“
6. Reboot to partiton „system b“
7. ‚dd‘ to partition „system a“ from „Download“
8. Cleanup (clean „Download“, delete pkg)

Root: OTA with modified firmware

- Preparation:
  - Download, decrypt, patch, encrypt pkg
  - Unprovision device (press reset key once)
- Retrieve token via discovery
  - Connect to open wifi network
  - Retrieve token, e.g. with python-mirobo discover
- Perform OTA update
  - Send *„milO.ota" command with own http server*
  - Wait until update was successful (~5 min)

Observations

- In provisioned state
  - APP-commands are not accepted, if Cloud-communication was not established
  - Block if cloudserver ip part of local subnet
    - DNS modification to local server does not work
- Firmware updates on STM32 MCU
  - STM is flashed thru the Application CPU
  - Firmware stored in rockrobo application folder

Secrets and device configurations

- Keys
  - Key (16 byte alpha-numeric)
    - Is used for cloud communication
    - Static, is not changed by update or provisioning
    - Example: „Abbb1deFGHijKlMN”
  - Token (16 byte alpha-numeric -> 32 byte asciihex)
    - Is used for app communication
    - Dynamic, is generated at provisioning (connecting to new wifi)
    - Example: „4a6d35524b513035494949 4a50535636”

Cloud configuration

- DID, Key
  - /mnt/default/device.conf
  - Timestamp of file equals to production date
- Token
  - /mnt/default/device.token
- Keys (cont.)
  - Vinda (16 byte uppercase letters + ASCII symbols)
    - Static
    - Used as some kind of secret in adbd and/or root password (console), exact use unknown
    - XOR(0x37)?
    - Example: „_BQQQ@S@D[BDMGGF”
      - xor(0x37) = “hufffwdwsluszppq”

Cloud protocol

- same payload for UDP and TCP stream
- encryption key depending of Cloud/App usage
- for unprovisioned devices:
  - while discovery: Token in plaintext in the checksum field

	Byte 0,1	Byte 2,3	Byte 4,5,6,7	Byte 8,9,A,B	Byte C,D,E,F
Header	Magic:2131	Lenght (hex)	00 00 00 00	DID (hex)	epoch (hex) (big endian)
Checksum	Md5sum[Header + Key(Cloud)/Token(App) + Data(if exists)]				
Data	Encrypted Data(if exists, e.g. if not Ping/Pong or Hello message) • token = for cloud: key; for app: token • key = md5sum(token) • iv = md5sum(key+token) • cipher = AES(key, AES.MODE_CBC, iv, padded plaintext)				