Wireless Goal Recognition in Foosball

Jonas Burster - 20165136

jburst16@student.aau.dk

xx.xx.2019

Contents

List of Figures

1 Introduction

maybe we should mention IAFoosball in introduction?
The Internet of Things (IoT) will change humanity in profound and unforeseeable ways[1]. According to a report by Cisco, between 2015 and 2020 the number of connected IoT devices is set to double from 25 billion to 50 billion[1]. This new revolution is driven by lower prices in manufacturing IoT devices as well as new inventions in radio communication and better processing power. This means more data than ever will be collected and automation will be the norm, rather than the exception. Industrial IoT (IIoT) and smart home are prime example for this. New factories are often operated by only a few or no workers at all. The former is called smart factories and the latter or lights-out factories[2]. In homes lights and temperature are increasingly controlled by an AI rather than the human living in it. This leads to higher productivity, better power management, more comfort and ther improvements. In most cases, this new technology is used to aid humans, but recent history has shown that IoT also brings many diverse challenges in technology, security and social life. Such as:?

In 2016 a botnet with over 600k infected devices, primarily IoT devices, overwhelmed several high-profile targets like the DNS provider Dyn with massive distributed denial-of-service (DDoS) attacks. This caused a temporary outage of their DNS servers making many webpages, among others Twitter, Spotify and Amazon, temporally inaccessible.

The sheer number of IoT devices also makes it important to think about their sustainability. Many IoT devices are cheap to manufacturer and thus only used for a short period of time. They are hard to update, because they usually lack a simple connection mechanism for updating and are placed in hard to reach places.

Often, this makes them throwaway products.

Finlly, some IoT devices are designed to aid humans in their homes and outdoor environment and are intended to analyze what they say, do and how their body behaves. This leads many professionals to the opinion that IoT, especially smart home and activity tracking are incompatible with privacy[3]. In the past many companies crossed the socially excepted line and had to roll back certain "features"[4]. Google recorded open WiFis during its Street View program(IoT related? i think google records all Bluetooth devices on their gHomes) and Amazons Echo, was and still is, recording all conversations. Studies have shown that for many consumers privacy is the major concern in smart home[4] and legislations have to be in place to check and balance the producers of IoT devices.

These four areas, technological improvements, security, sustainability and privacy, are the four areas shaping the development of IoT and the main drivers for many technological as well as architectural decisions in the development process of the product developed as part of this project. The reminder or this report is structured as follows, ***
Maybe end line could be something along the lines of, The reminder of this report, is exploring a small/big/(some measurement) of these areas

1.1 Motivation

In this project we develop a new wireless goal for foosball tables for the IAFoosball project. We already developed a simple goal detecting goal, but it was not portable to other tables. The setup was table specific and wired up in such a way that was not scalable nor future proven. In this project, we would like to improve these shortcomings. The goal is the central part of the table. Making it scalable and

easy to maintain is very important for the IAFoosball project going forward.

The backend software to manage user data and a primary architecture was part of a previous project and is shortly discussed in the next chapter. However, the choice of our backend software heavily influenced the software choice and automation flow. In our vision the table is a special, but integrated, part of our server architecture. This vision is shortly explained as orchestration in IoT is not part of the curriculum.

In order to make our solution attractive to bars and companies alike the solution neeeds to be easy to implement and manage. The resulting research question for this project is as follow:

What is the optimal design for a scalable wireless foosball goal?

We will answer the question based on the four aspects: technology, security, sustainability and privacy.

1.2 Delimitations

Due to time and equipment limitations we will focus on the technology in this report. Also, the findings from our tests are useful as a general reference point when discussing the strength and weaknesses of competing solutions. But as the tests do not follow strict academic rules, these findings are not statistical significant.
We also limit the main scope of this report to how we send and receive data from the goal to its controller. We will show the entire architecture but it is not part of the requirements and often not analyze or explained as throughly as the goal sensors.
Finally, we did not gather input from experts. This is something we would have wished to do, but was not possible due to time constraints.

2 IAFoosball

IAFoosball is the name of the project and startup we developed in the previous semester. It is meant to make foosball more interactive, to show and share statistics, and make it more sociable by using new technologies. The planned services include global and private rankings, table finder, friends, automatic tournaments and more. Because it is part of an university project, we used the latest and greatest technology. The front end is written in Flutter and pReact, the back end is separated in containerized microservices written in Go and all communication is done through gRPC, which uses protobuffs and HTTP/2. Figure 1 shows this architecture.

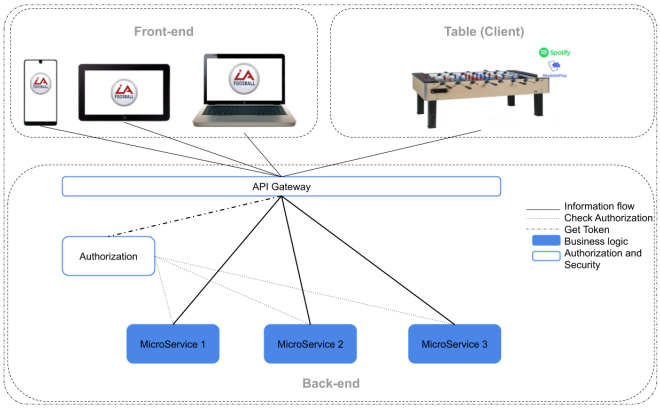

Figure 1: The current IAFoosball architecture

The foosball table in ***PLEASE UPLOAD PICTURE***is the development version and has all features. It includes, speakers with Spotify integration, LED lights, automatic ball release and a tablet. However, the software and hardware used on the table does not life up to the software standards at IAFoosball. The software needs to be updated locally without a CI/CD (continous integration and continous delivery) pipeline and is attached physically to the IoT gateway, the raspberry pi. This makes it very hard to scale and not mainstream suitable. To

make IAFoosball more compelling for people and business which just want goal counting, and possibly ball speed measurments, we wanted to cut down on features and instead concentrate on quality.

2.1 Goal Design

The goal design is the primary part of this report and the most lacking part in the former project. In figure 2 the old electronic wiring is shown.

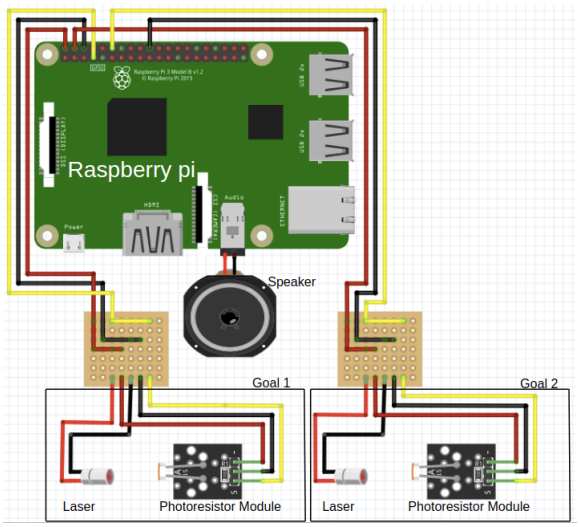

Figure 2: The old IAFoosball goal

The sensors were directly connected to Raspberry Pi, and a simple javascript application registered the goals and sent them to a remote server. This required a power outlet and a Raspberry Pi on each table. It also required cables from each goal to the Raspberry Pi. This approach did not scale well and in a multi-table setup also wasted money as each table needed its own Raspberry Pi. We re-thought the design, making it more versatile, easier to integrate and manage and also cheaper for multiple tables.

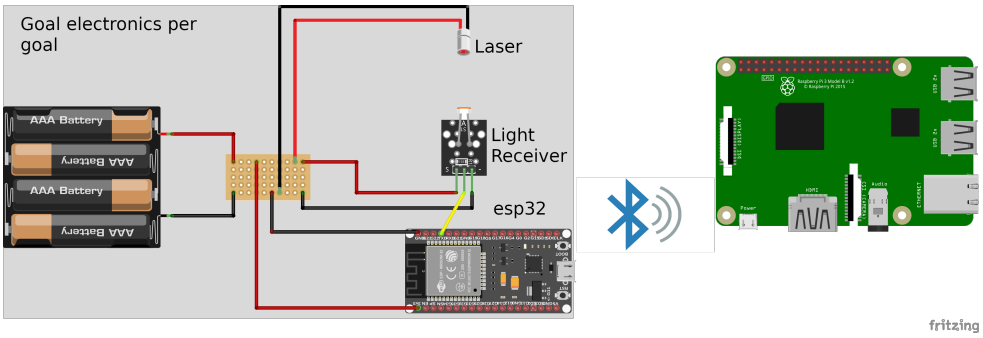

Figure 3: The new IAFoosball goal

Figure 3 shows this new setup. Marked in grey are the components used for each goal, a battery (we use a lithium-ion battery), one laser, one light sensor and one esp32. To measure goal speed we would need to have to lasers and two sensors. The esp32 is the component doing the computation and sending data to a Raspberry Pi via bluetooth low energy (BLE). It is powered by a battery pack and only active when the need arises.

3 Requirements

The requirements are split into functional and non-functional requirements, where the former are definitions of what the system is supposed to do and the latter requirments describing how the system is supposed to be. The respective tables are included at the end of the chapter to enhance readability.

The functional requirements are classified with the MoSCoW method into Must, Should, Could, and Would and are listed in section 3[1]. The musts reflect the requirements for our project from this course. Our objective is to create a functional wireless automatic goal registration device driven by a battery and having a low

[1] Class stands for the MoSCoW classifier.

maintenance cost. Power saving mechanisms are thus a top priority and tested in the respective chapter (add reference). The overall architecture is also very important for mangement and fault tolerance. But as the topics are not an essential part of the curriculum they are not explained extensively. Finally, "security in IoT devices is either neglected or treated as an afterthought"[5] by many manufacturers. Security is part of the curriculum, but also something encompassing communication as well as the system design. We thus split security into two areas, architecture security and encryption, changing a system is often much harder than upgrading the encryption. Encryption is "only" ranked as a should, as a functioning prototype is more important.

The criteria for the non-functional requirements are shown in section 3. In contrast to many consumer facing technologies usability is not part of the requirements, as the enduser should have as little interaction with the system as possible and physical maintenance is only carried out by professionals[2].

The non-functional requirements cover the areas: Reliability, Robustness, Portability, Maintainability and Efficiency, again, focusing on this course curriculum. How to make the device as power efficient as possible will be the biggest discussion point, again.

[2]They would still be nice, but due to time limitations we excluded them.

Functional Requirements

ID	Name	Class	Description
100	Register goals	Must	The system must be able to automatically register goals
101	Transmit goals	Must	The system must be able to automatically transmit goal information to the edge device.
102	OTA updates	Should	The microcontroller should support over the air (OTA) updates
103	Low energy transmission	Must	The microcontroller and its transmission partner must be able to communicate via a low energy transmission protocol
104	Power modes	Must	The microcontroller must be able to support different power mode in which it can operate to save power during unused periods.
105	Wake up from sensor	Must	The microcontroller must be able to wake up from an external sensor input.
106	Wake up from timeout	Must	The microcontroller must be able to wake up after a predefined timeout.
107	Transmission range	Must	The transmission range must be larger than 5 meters without blockage.
108	Pin reads	Must	The microcontroller must support reading from input pins.
109	Centralized configuration	Should	All configurations should be synchronized across the nodes in the system.
110	Node fault tolerance	Should	The state of each node should be synchronized in the system, so that in case of an error, the last operable state can be recovered
111	Adjustable Goals setup	Should	The goals setup need to able to adjust in their width to cover a variety of different goal sizes

Functional Requirements

ID	Name	Class	Description
112	Containers (excl microc.)	Should	All applications, excluding the microcontroller, should run inside containers.
113	Containers	Would	The microcontroller code runs inside containers as well.
114	Single binaries	Could	All applications should be packaged in one binary.
115	Edge storage	Would	The edge part should be able to count goals and syncronize with user devices without the Internet.
116	Encryption	Should	All data should be encrypted with AES-128 or better.
117	Verified updates	Would	The microcontroller would only install signed and verified updates.

Non-Functional Requirements

Area	ID	Name	Description
Reliability	100	Downtime	The downtime due needs to be less once per month.
	101	Recording failures	Less than 5 percent incorrect readings.
Robustness	200	Interference	The system should be able to deal with at least 10 bluetooth enabled devices nearby.
	201	Transmission failures	There should not be more 0.01 percent of transmission failures, including all reasons.
	203	Incorrect data	There should not be more 0.0001 percent of wrong data transmission.
	202	Crashes	In case of a crash, the software must be able to recover automatically.
Portability	300	Supported platforms	The application needs to supported on a wide variety of IoT devices for future changes.
Maintainability	400	Updates	The software needs to be updatable over the air (OTA).
	401	Centralization	As long as a table is connected to the Internet (also indirectly through the raspberry pi), all updates must be available through a central point.
	402	Bug fixes	All bugs need to be addressed latest 6 months after discovery
Efficiency	500	Battery life	The system needs to able to life on a single battery charge for at least two weeks with no more than 20 games per day.
	501	Battery replacement	The battery needs to be able to charge to above 80 percent of its original value after 2 years, with 25 recharges a year.

4 System Architecture

The system architecture defines the structure and behaviour of a system. IAFoosball is a solution for an interactive user experience for foosball. It thus compromises a cloud part, an edge part and a constrained device (IoT part). This is shown in figure 4. We will focus on the edge and IoT part and only brievly touch on the cloud part for completness.

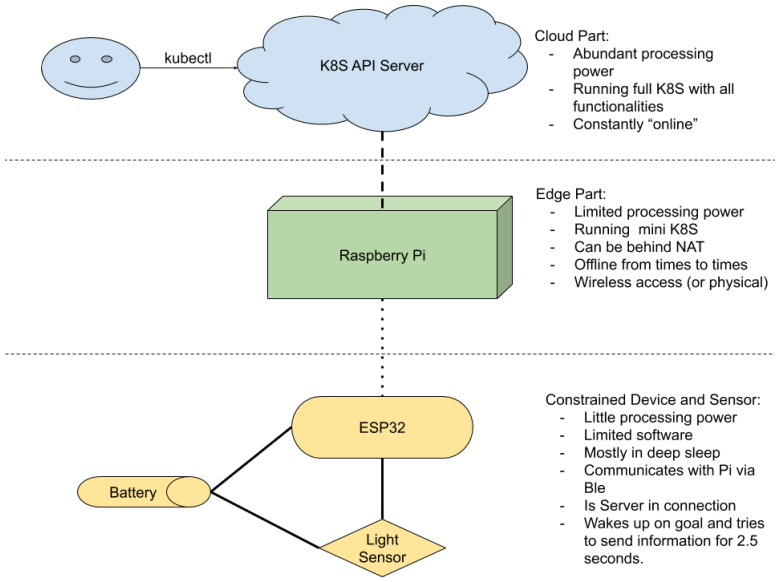

Figure 4: The System Architecture

The cloud is where all user relevant data is stored and where the system is administered. It runs on Kubernetes, which abstracts the physical hardware programs run on and puts them into pods. Pods are smalles unit in Kubernetes and are isolated into their own namespace and cgroup and can contain multiple containers. Unlike virtual machine, it does not create any overhead at runtime for containers, but t offers the possibility to fully orchestrate the pods. This means, pods can restarted if they are not behaving correctly, replicas can be turned up or down depending on the system load, it offers automated rollouts and the Kubernetes itself is highly

extensable.

Kubernetes was developed as a cloud platform, but as edge devices become more powerful and do more processing of data, especially executing business logic, managin them is becoming an increasingly popular research topic. Processing at the edge is called edge (or fog) computing and has many challenges. The edge devices can have very limitted power, they can be offline from time to time, have a slow Internet connection and be behind a NAT. Kubernetes was not designed to be used on the edge, but because its extension model, engineers are now working exactly on that. This means, an edge device can be just another node inside network, treated differently than other nodes though, but still containing all the advantages Kubernetes offers.

Last but most importantly, the IoT part is where the actual sensing of the data happens. This device is so processing power restricted that it can not run Kubernetes, but updates for it can be build inside a container running either in the cloud or on the raspberry and pushed to the microcontroller via a OTM update. This makes updating multiple devices feasable and ensures that updates, especially security updates, can be rolled out quickly.

The actual architecture of the goal is compromised of the esp32, a battery and the laser and a photon resistor. When a goal is scored, meaning the laser does not shine on the resistor, a goal is registered. The esp32 then opens a ble server and notifies the edge device of the goal. How this is exactly done is discussed after the testing chapter. If the rpi acknowledges the goal, the Laser always on???

5 Measurements and Tests

This chapter covers the power drain from the sensor, laser and the esp32, inlcuding deepsleep and bluetooth, to determine the optimal setup. The measurement or test

setup is explained in each respective section. The tests are guiding points and are not intended for scientific or statisitcal significance. Their goal is to provide an order of magnitude of the power consumption of the individual components to guide future architectural decisions. This chapter will first look at the laser and sensor measurments and afterwards at the power consumption of normal/idle and deepsleep and then how much a wake up from deepsleep costs and transmitting data.

5.1 Laser

We measured the laser on a power source directly connected to the esp32 which has an output rating of $4.695V$ with a max current of $80mA$. The laser consumed on average $25mA$ and thus the power consumption is

$$4.695V * 25mA = 0.117W$$

which is, as expected, fairly high. In a setup with to lasers per goal, this could be a realy problem

5.2 Sensor

Next, we measured the power consumption of the sensor, again powered by the esp32, so $4.695V$ with a max current of $80mA$. It consumed on average $3.5mA$ when the laser was hitting the sensor and $3.8mA$ when it was not. As the default state is hitting and only goals, so very short periods, are not hitting, the power consumption is

$$4.695V * 3.5mA = 0.016W.$$

This is inline with the expectations, as the resistance drops when the laser hits the sensor. To summarize, together the sensor and laser consume around

$$0.016W + 0.117W = 0.133W$$

in idle and a little bit more when a goal is scored. Thus the power consumption in one day is

$$0.133W * 24 = 3.19Whr$$

This is quite a lot especially running on a battery with only

$$0.133W * 24 = 12Whr$$

5.3 ESP32 Modes and BLE

The esp32 development board from wemos consumes around $43mA$ in idle and $0.014mA$ in deepsleep. This is much higher than esp32 wroom board which is supposed to only consume $20mA$ in idle and $0.01mA$ in deepsleep[6]. Figure 5 shows both microcontrollers side by side.

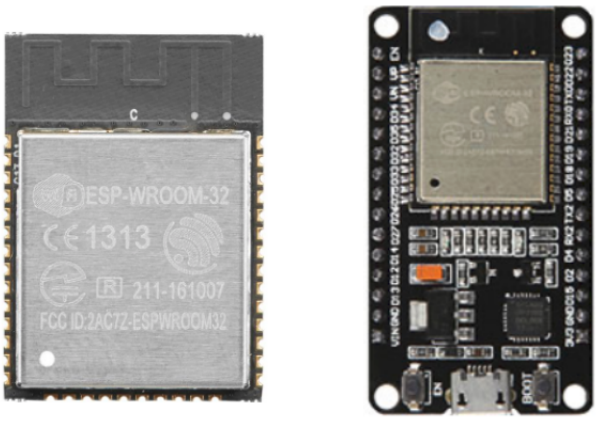

Figure 5: ESP32 wroom (left) and Wemos development board.

The development board has a microusb controller and additional current controls and a power LED. These extra components add up to the significant power consumption difference. For the production we thus plan on using a esp32 Wroom and soldier the connections ourselfs. Thus, instead of using our numbers we will

use the numbers from the esp32 wroom controller[6].

The power consumption also changes during boot time and during data transmission. In the former, the processor consumes a lot of power, while in the latter, the radio chip, including antenna, consumes the majority of extra power. Unfortuately, we were not able to get a esp32 wroom in time for the tests, so we have to infere values from the esp32 wemos development board.

In our tests, waking up from deepsleep took around $250ms$ and increased current consumption to about $80mA$[3]. Especially, the wake up time seems quite slow and we would like to test this further and search for wake up optimizations in further research. During the initialization of the bluetooth radio power consumption was similar to the wake up, again showing that the processor is tasked more heavily. Sending and receiving of data, the power consumption increased to $50mA$ but for such short time periods that in our calculation we omit sending information. Unfortuately, we did not have the time to test how much power it cost to act as a slave and search for ble beacons and receive data.

To summarize each wake up costs

$$0.08A * 5V * 1h = 0.4Wh$$

per hour so for a startup time of $1/3s$ that constitutes to

$$0.4Wh/(3600/(1/3)) = 0.000037W = 0.037mW$$

Initializing the bluetooth radio is fast at around $1/10s$ and consumes

$$0.08A * 5V * 1h = 0.4Wh0.4Wh/(3600/(1/10)) = 0.000011W = 0.011mW$$

[3]This was done via eye measurment with a lot of fluctations, so the numbers are really only approximates.

5.4 Summary

Together the laser and sensor consume $0.133W$. The esp32 consumes $0.01mA$ in deepsleep and $20mA$ in idle. It spikes at around $80mA$ for initialization periods which total 4.3/10 and thus each start costs $0.048mW$.

References

[1] D. Evans, "The internet of things: How the next evolution of the internet is changing everything", *Cisco Internet Business Solutions Group (IBSG)*, vol. 1, pp. 1–11, Jan. 2011.

[2] J. P. Tomás, *Smart manufacturing vs. lights out manufacturing*, `https://enterpriseiotinsights.com/20170616/channels/fundamentals/20170616channelsfundar lights-out-manufacturing-tag23-tag99`, (Accessed on 05/27/2019), Jun. 2016.

[3] S. Morrow, *5 reasons privacy and iot are incompatible | iot for all*, `https://www.iotforall.com/five-reasons-privacy-iot-incompatible/`, (Accessed on 05/17/2019), Oct. 2018.

[4] R. MARVIN, *Privacy tops list of consumer smart home concerns - pcmag uk*, `https://uk.pcmag.com/nest-secure/119897/privacy-tops-list-of-consumer-smart-home-concerns`, (Accessed on 05/17/2019), Mar. 2019.

[5] J. Wurm, K. Hoang, O. Arias, A.-R. Sadeghi, and Y. Jin, "Security analysis on consumer and industrial iot devices", in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, IEEE, 2016, pp. 519–524.

[6] *Insight into esp32 sleep modes & their power consumption*, `https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/`, (Accessed on 05/28/2019).