Reliable IoT Systems for Improving Quality Of Life Through The Exploitation of Cloud, Mobile and BLE Low Energy Based Technologies. Case Study: Battery Charge Protect

***Abstract*—Connecting various smart objects within an intelligent ecosystem provides high capability of developing and integrating mobile, cloud and embedded device communications based on existing internet infrastructure. This has been the trend of Internet of Things (IoT), addressing health, quality of life, smart cities. In this paper we are a focused on similar aspects with a proper case study in an embedded system, used in protecting batteries of mobile devices. Cloud/Mobile applications interconnected with embedded devices shall be presented. The research undertaken present not just the technological innovation, exploiting state of the art technology, but also the security benefits and prolongation of battery life. The system has been fully developed, tested and it's benefits evaluated throughout the paper. We have been able to integrate and provide a real life case study adopting this integrated architecture to be reused for future implementations.**

***Index Terms*—Internet-of-Things, Smart Devices, Fall detection, Energy eciency, Wearable devices**

I. Introduction

Internet of Things (IoT) concept and paradigm brings together existing internet infrastructure with everyday embedded devices such as smartphones or other micro-controllers in a common playground to improve people everyday life and provide smart innovative services. Information technology relying on Internet has widely grown during the past decades [1]. Thus all services and information are exchanged on World Wide Web (www). However a new concept arises with the need to provide interconnection not just between web browsers and cloud based servers / clusters with robust ISO protocols, but also between smart embedded devices. The latter might rely on independent protocols with respect to the ones used for interconnecting Web applications. The main goal of the IoT is to bring together both paradigms of the 21st Century to collaborate in creating larger grids of intelligent services and devices. Thus a mobile / cloud / embedded device interconnection through existing state of the art technologies shall be presented along this paper work. A development approach and methodology will be described and adopted to real case study addressing primarily health related issues. Thus a system monitoring the battery through a mobile application and BLE device is unfolded in the coming section. The main purpose of the Charge Protect Application is to control, predict and suggest the battery level of charge

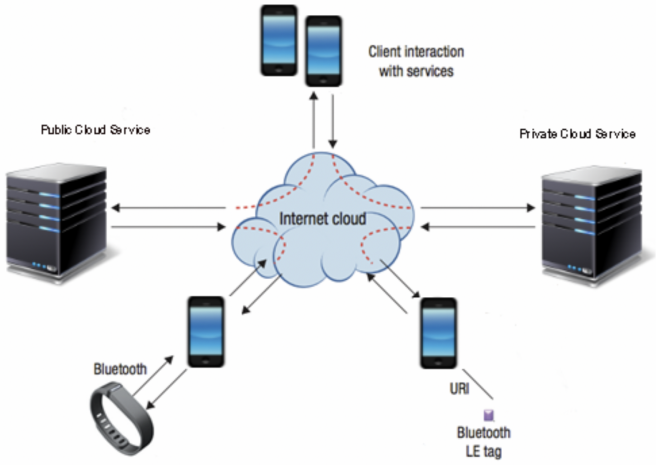


Fig. 1. Adopted architecture to interconnect the existing cloud infrastructure with embedded devices relying on Bluetooth Low Energy (BLE) communication protocol [1]

(SoC), providing not just useful tips but also security benefits which could incur from battery issues. After results have been gathered in the last section outcomes and challenges related to technological restrictions and decisions undertaken will be evaluated together with future proposals of improving existing approaches.

II. Background

World Wide Web services are obviously the primary source and model of interconnection between services and information. Several existing embedded devices can become part, interact or controlled from existing web technologies. This would result into a combination between the previous two thus deriving the concept of Physical Web = Web Technology + IoT [2]. Figure 1 represents an overview of the possible physical web architecture to be adopted.

Existing concepts in web/cloud protocols will be extensively exploited along having a stable and robust infrastructure in storing data, sharing/retrieving information as well as guaranteeing high availability and scalability. Thus, URI (Unique Resource Identifiers) and RESTful API (REST standing for

Representational State Transfer) will be part of the system architectures. Moreover further protocols are required in order to fill the communication gap between smartphones and embedded devices relying on state of the art micro-controllers. One of the largely adopted technology is the one based on Nordic Semiconductors, which has a built-in BLE embedded inside its System on Chip (SoC) micro-controllers. Also the Bluetooth communication protocol relies on Generic Access Profile (GAP) and Generic Attribute Profile. GAP covers the usage model of the lower-level radio protocols to define roles, procedures, and modes that allow devices to broadcast data, discover devices, establish connections, manage connections, and negotiate security levels, GAP is, in essence, the topmost control layer of BLE. This profile is mandatory for all BLE devices, and all must comply with it. GATT deals with data exchange in BLE, GATT defines a basic data model and procedures to allow devices to discover, read, write, and push data elements between them. It is, in essence, the topmost data layer of BLE. Both protocols assure a stable communication of data and services with SmartPhones having BLE technology built-in. (Integrated from Bluetooth vs. 4.0). This has result the most promising standard starting from 2010.

III. Literature Review and Motivation

The paper addresses not just concerns related to technology but also the health benefits related to the exploitation of BLE embedded devices. Many publications concerning UV (Ultraviolet) dosage intake have been a major concern related to skin health issues. One of the major risk being skin cancer [...]. While several articles during the past five years have been published related to development of embedded devices based on different micro-controller technologies operating in different settings including health benefits [3, 4, 5]. One of the most promising seems to be Nordic Semiconductors where BLE integration as a communication protocol has been made easy [6]. Several authors have presented the possibility of integrating into the Physical Web approach Cloud/Mobile and Embedded Devices, with high proficiency, security, availability and easiness of usage [7]. After analyzing charge current as function of charge Voltage. We made a simple adjustable charger circuit to analyze how different phones responded to different charge voltages. The test charger had adjustable output voltage from 4.0V to 5.3V. Max current 2A I had five different devices to test; HTC Desire, Samsung Galaxy S8, iPhone 6, iPad 2, iPhone 7). Had to spend some time to analyze how the Phones could detect the max current the different charges could deliver. Found that USB data lines D- and D+ are used to tell the charged device how many Ampere they can deliver. There are different standards (Note that iPhone do not follow these standards, but have made their own) how the phone can identify the chargers max charge current by using resistor networks in the charger to set different voltages on D- and D+ signals. See below for some different D+/D- configurations. Samsung used shorted D- to D+ and voltage set to 1.2V for 1A charger and Logic Low (must test) for 2A charger. iPhone has D+=2V and D-=2.7V for 1A and D+=2.7V and D-= 2.0V for 2A charger: Each phone was connected to the charger with correct D-/D+ voltage for max current charge and charge current was recorded for each 100mV starting at 5.3V and down to 4.0V. Table I. shows the Charge Current as function of Charge Voltage, while Figure 2 shows the corresponding plotting.

TABLE I
Component-based software engineering of the system

	Phones and init charge level [%]							
	20 %	80 %	51 %	24 %	16 %	54 %	16 %	74 %
Voltage	HTC Desire [mA]	Galaxy S3 [mA]	Galaxy S5-1 [mA]	Galaxy S5-2 [mA]	Galaxy S6 [mA]	iPhone 4 [mA]	iPhone S6 [mA]	iPad [mA]
5,30						970	1020	1530
5,20	904	950	1256	950	1193	895	1020	1412
5,10	899	942	1193	950	1193	828	1018	1300
5,00	890	670	1045	890	1196	758	1021	1185
4,90	870	484	947	754	899	690	908	1048
4,80	811	483	753	650	714	626	814	903
4,70	733	299	558	521	510	551	706	762
4,60	650	116	459	420	413	418	606	620
4,50	576	113	313	310	314	257	504	501
4,40	518	111	202	182	180	0	418	381
4,30	448	0	0	18	0	0	339	362
4,20	380	0	0	0	0	0	209	328
4,10	303	0	0	0	0	0	166	50
4,00	223	0	0	0	0	0	88	50

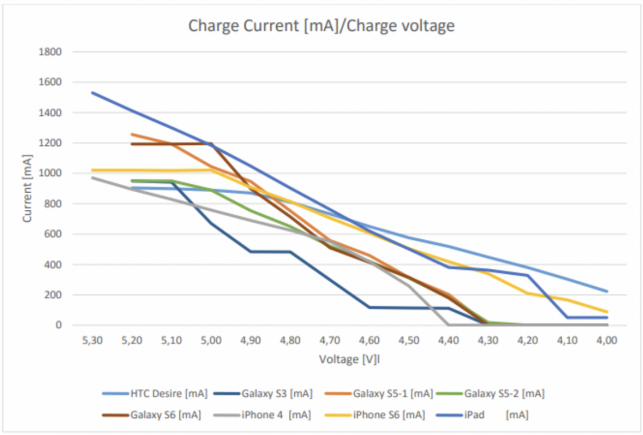


Fig. 2. Plot showing charge current as function of Charge VoltagePlot showing charge current as function of Charge Voltage

Initially we had an idea using a current limiter for low current charging for Long life mode, but after analyzing the charge current, I think the most optimal design will be to set low charge to 4.6V, medium charge to 4.8V and 5V for max current (what the charger can deliver). 4.6V will set max current to 600mA and a linear LDO regulator could be used instead of a Buck step down DC-DC converter. LDO are cheaper, fewer components and we will eliminate EMI emission issues. Maximum power loss using LDO will be 0.4V x 0.6A= 0.24W. Well below what could cause Dongle thermal issues.

A. Charge Current Detection Coding

Device makers being a competitive lot, theres no one standard negotiation its a bit like having to speak six languages. While the USB forum released a generic signature standard, other manufacturers came up with their own signatures and in the end, there are at least six D+/D- signatures in the wild:

- 2.0V/2.0V low power (500mA)
- 2.0V/2.7V Apple iPhone (1000mA/5-watt)
- 2.7V/2.0V Apple iPad (2100mA/10-watt)
- 2.7V/2.7V 12-watt (2400mA, possibly used by Blackberry)
- D+/D- shorted together USB-IF BC 1.2 standard
- 1.2V/1.2V Samsung devices

These days, resistance-based voltage sensing options like the first four are described as legacy modes and all new devices we believe use chip-based detection.

In the following sections we will describe the methodology and the case study to address the problem.

B. Methodology

Development of an integrated environment of Cloud / Mobile and Embedded wearable devices consists in the following steps:

1) Development of the Wearable Device (Dongle)
2) Firmware Development providing the communication protocol between the mobile
3) Mobile App Development (Android / iOS)
4) Google Cloud Communication

The system is general and abstract set of hardware and software making use of the common software engineering concept CBSE (Component-based software engineering) shown in Figure 3. The benefits of such an approach is that each component is reusable and highly configurable. The system architecture can be described as an onion with a core containing the abstract and general building blocks while more outer layers build upon the inner layers becomes more and more concrete and specific the further out in the onions layers.

The device micro-controller is based on Nordic Semiconductor technology with nRF51 Development Kit, from which a firmware development in C programming language is made possible. Charge Guard is basically a charge limiter for your phone battery. The intended purpose of the product is to limit the charge level of the phone battery to a certain level in order to prolong battery lifetime. The reason for why this works is that the higher the voltage of the Lithium-ion battery gets; the more wear the cells will be exposed to. By preventing the cell voltage of getting to its maximum charged level the wear on the battery cells are reduced and the battery lifetime will be prolonged. Figure 4 shows the intercommunication intended between the Charger/Dongle/SmartPhone.

A Client / Server (C/S) Architecture is adopted for broadcasting services and exchanging data between the Mobile Application and the Wearable Device. Firmware acts primarily as the Server while the mobile application developed in Android/iOS etc. shall act as a client communicating with

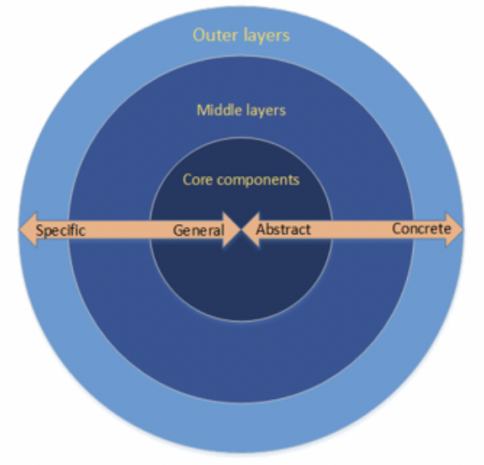


Fig. 3. Component-based software engineering of the system

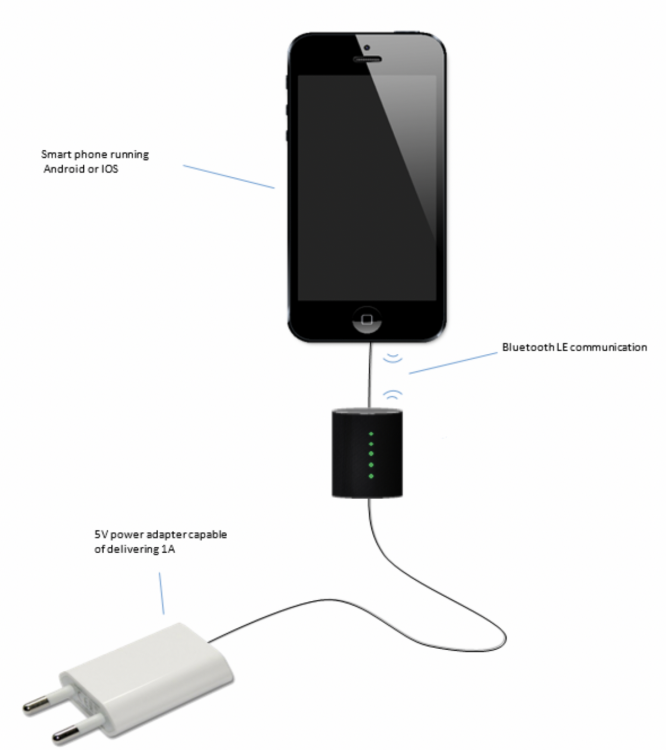


Fig. 4. Dongle interconnection between charger and device

the services offered from the server based on the GAP/GATT protocol. Further status of the current charge is updated on the Google Cloud from the mobile application. The overall architecture is described in Figure 5.

IV. Case Study

Charge Guard is basically a charge limiter for your phone battery. The intended purpose of the product is to limit the charge level of the phone battery to a certain level in order to prolong battery lifetime. The reason for why this works is

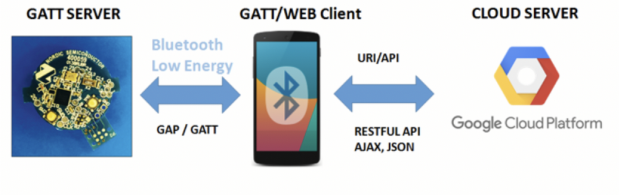


Fig. 5. IoT System Architecture

that the higher the voltage of the Lithium-ion battery gets; the more wear the cells will be exposed to. By preventing the cell voltage of getting to its maximum charged level the wear on the battery cells are reduced and the battery lifetime will be prolonged.

It will also give user an alarm when battery reach the predefined low battery level to start charging. Based on user setting the Charge Guard will also enable fast (high current charge) or slow (low current charge). Slow charge will charge with a lower current and prevent Battery from reaching high temperature that also can shorten the battery life. The last function is to monitor Battery temperature and charge current. It shall also activate multiple alarms (Vibration, sound, light) for user to act in order to avoid any hazard and turn off the charge current by internal switches. There are two main domains within the charge guard domain; The phone app and the charge limiting dongle itself. The respective features of each of them is described in the sections below.

A. *HW Functionality Description*

The CG unit is a Battery charging protection device designed to prevent overcharging batteries in order to extend the battery lifetime. The CG is controlled by an App installed on the unit to be charged. The CG/App has several built in functions:

1) Fast or slow charge mode. The App has a user selectable charge speed (slow/fast) switch.
2) Battery under/over charge mode. The App has user defined lower and upper charge limits. Low setting will initiate start charge message to the user. Upper charge limit is defined by user and will turn of power to the Phone/PAD when limit is reached. A predefined default setting is set, but can easily be adjusted by user.
3) Charger Emulation. The CG will read charger signature set on USB d-/d+ lines and convert the levels to match the Phone/Pad brand and model for optimal charge current. This function will enable phones of different brands to use charger from other brands without risk for damaging charger or phone/pad.
4) Safety function will monitor voltages from charger and to phone in addition to charge current and temperature. If any of these are outside defined levels the Device will immediately abort the charging sequence and activate multiple alarms as visual, audible and vibration.
5) Allow normal charging current provided by manufacturer and charger specification.

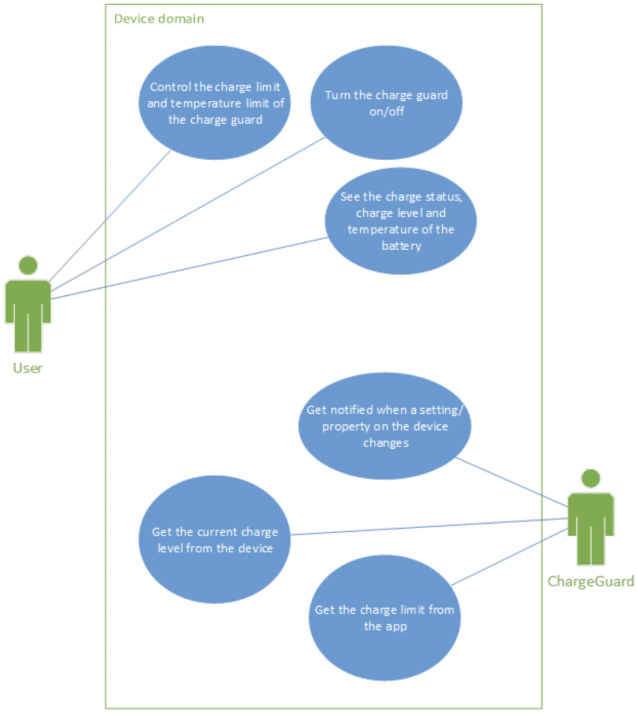


Fig. 6. System usability between the Dongle and the User

6) Allow low/medium charge current by setting charge voltage to 4.6V or 4.8V
7) Monitor Dongle temperature, charge voltage and current.
8) Send Alarm message if Dongle temp reach 60 oC
9) Emulate other brand chargers
10) The Charge guard shall be connected to the Charger and with cable to the device/phone/Pad.
11) Connectors USB-A is a universal solution

The usability of the system is represented in Figure 6

The dongle operates based on the implementation of a firmware dedicated exclusively to it. While based on the architecture presented in the methodology we implemented the mobile application in Android. The standard interface is through USB-A cable. While communication interfaces are between the charging dongle and the phone. BLE communication is established between dongle/Device pairing by Dongle Output Voltage cycling through a sequence changing output voltage between 4.6 4.8 5V to confirm correct dongle connected to the Device. If output voltage does not change according the programmed sequence the Paring must repeat this HW handshake for each dongle until match is detected. The Dongle includes programmable HW voltage design to supply 0V, 4.6V, 4.8V and 5V (the Charger output voltage) to the Charging Device (Phone/Pad or eq). It also monitors Charger Voltage and Voltage fed to Phone to prevent any Hazards by turning off output Voltage when abnormal voltage/current is detected. Current measurement is added to the Output voltage for safety and user information. The Current sense must read current up to min 3A. Simulation in Fig. shows Current Sense

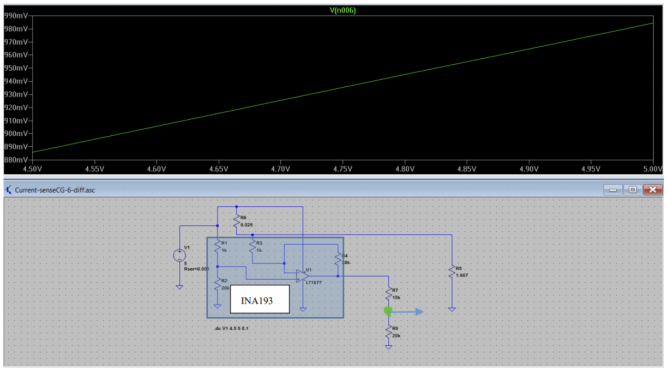


Fig. 7. Simulations for Current Sense design using INA193-198 Current sense Monitor for High side current sense

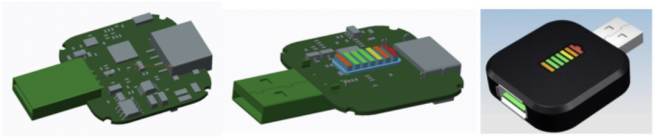


Fig. 8. Hardware industrial design of the dongle

design using INA193-198 Current sense Monitor for High side current sense. Select INA193/196 with Gain=20 and divide by 10k/20k to give 3A=1V output to nRF ADC, Figure 7

B. HW Detail Design

The hardware industrial PCB design of the dongle is represented in Figure 8 where physical dimensions are:

1) Width: 35 mm
2) Depth: 60 mm (including connectors)
3) Height: 20 mm

The Dongle has LED based color coded charge level. The Tri-LED is also used for illumination (Alarm flashing and low ambient light lamp). Six LED shall be included for Battery status level to user covering the range from 0 to 100%. The Battery indicator have the color coding as in Table II.

While interfaces of communication between the charger and Dongle are shown in Figure 9

The Table III represents the electrical parameters for the connectors.

C. HW/FW functional description

Table IV shows the functional dependencies between Hardware/Firmware developed for the dongle, while communication requirements are described in Table V

Development of the firmware is based on Nordic Semiconductors with nRF51 development kit [reference] based on C programming language. In Figure 10 can be observed the programming board interconnected to the dongle prototype (to the right) so that the developed firmware can be deployed on it. The firmware will execute on a microcontroller in an embedded system without the use of an operating system. The microcontroller chosen for the project is a Texas Instruments (CC2564) SOC (System on Chip) BLE enabled processor.

After being programmed the dongle allows for charge voltage controlled programming as described in Table VI

TABLE II
ELECTRICAL INTERFACES FOR CONNECTORS

Charge Level [%]	LED Color
100	RED
80	GREEN
60	ORANGE
40	GREEN
20	GREEN
0	RED

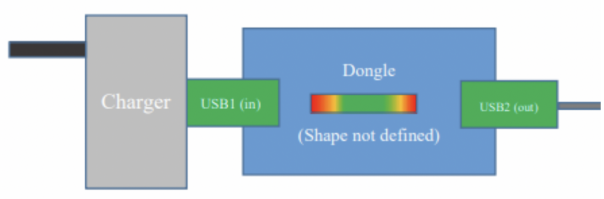
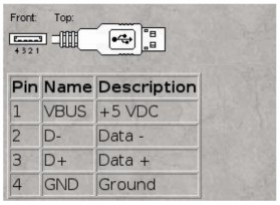


Fig. 9. Communication interfaces between Dongle and Charger

TABLE III
ELECTRICAL INTERFACES FOR CONNECTORS

Connector Description	Des	Type	Comments
Programming Connector	J1	10 pin 1.27mm	Dual row
USB Charger side	USB1	USB-A Male	Input
USB Phone/Tablet Side	USB2	USB-A Female	Output



Pin	Name	Description
1	VBUS	+5 VDC
2	D-	Data -
3	D+	Data +
4	GND	Ground

D. Mobile and Cloud Functionality Description

Using the phone app user interface the user must be able to set a level of charge that the phone should alert the user

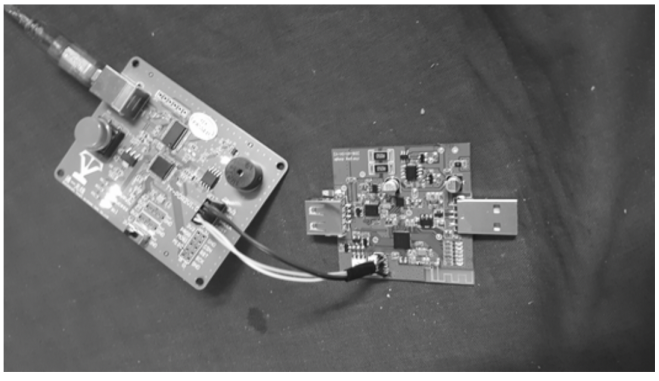


Fig. 10. Programming the device with Nordic Semiconductor Development Kit

TABLE IV
HW/FW DEPENDENCIES

Signal	Function	U1 - pin	I/O	Signal level active	Comments
T1	Turn on Max Voltage	28	O	High	Set output voltage to charger output voltage
T2	Turn on 4.6V output Voltage	21	O	Low	
T2	Turn on 4.8V output Voltage	21	O	High	
ENABLE	LDO Enable signal	22	O	High	
LED1	LED RD or Tri Color	43	O-PWM	Low	LED not defined
LED2	LED OR or Tri Color	44	O-PWM	Low	LED not defined
LED3	LED OR or Tri Color	45	O-PWM	Low	LED not defined
LED4	LED OR or Tri Color	46	O-PWM	Low	LED not defined
LED5	LED GN or Tri Color	47	O-PWM	Low	LED not defined
LED6	LED GN or Tri Color	48	O-PWM	Low	LED not defined
LED9	Tri Color LED	18, 19, 20	O-PWM	High	Transistor inverter
PWM d-	Opamp	15	O	PWM	
PWM d+	Opamp	16	O	PWM	
MUX Addr.A	HCT4053	25	O	X,X,X	Binary Addressing
MUX Addr B	HCT4053	26	O	X,X,X	Binary Addressing
MUX Addr C	HCT4053	27	O	X,X,X	Binary Addressing
Current Sense	Read Current to Phone/Tablet	7	I	A/D 0-1.2V	Analogue input
Uin	Input Voltage	9	I	A/D 0-1.2V	Analogue input
Uout	Output Voltage	8	I	A/D 0-1.2V	Analogue input
USB D-	Input to nRF for Charger ID reading	6	I/O	A/A 0-5V	Used to decode charger and emulate diff types
USB D+	Input to nRF for Charger ID reading	5	I/O	A/A 0-5V	Used to decode charger and emulate diff types
SWDIO	Programming Data In	23	I	Data	Flash programming
SWDCLK	Programming Clock	24	I	Clk	Flash programming
USB Shield GND	USB Shield not connected with signal GND		-		Follow USB spec for PCB design. Connect USB1 Shield to USB2 Shield only
X-TAL		36,37	I/O		MCU clock
BLE antenna		30-32	I/O		

of when it is reached. When connected to a Charge Guard charging dongle the charge limit should be reflected in the hardware dongle.

The app must continuously monitor the level of charge in order to check whether the charge level is reached triggering a phone alarm/notification. If the app is connected to a Charge Guard dongle must be notified on order to stop the charging by cutting the power to the device.

The app will let the user determine a limit for max charging current and temperature limit (only in cases where the phone operating system supports retrieving it).

So in summary the mobile application associated to the system serves following purposes:

- Limit the charge to a certain user specified level
- Battery level monitoring
- Let the user specify the max charge limit
- Present a user friendly User Interface (UI)

The use cases representing the system usability are represented in Figure 11

The operating system requirements for the devices are:

- Android Minimum supported version greater or equal 4.1.x API version 16
- GATT server must run in background

The mobile application interfaces are represented in Figure 12

While Google Cloud Application is running on Django framework 1.11 and Python 2.7. The purpose of the cloud system is to keep track of the battery status sent from the

TABLE V
HW/FW COMMUNICATION

Function	*APP SW*	*Dongle FW*	*Signal levels*	*A/D conv. input*	*Comment*
BLE Pairing	Send Uout Sequence to CG and pair when HW responds to request	Toggle Uout as defined in order to identify connected device until match.	4.6 to 5VDC		
Charger Signature ID (d+/d-)	Send Phone ID to CG to set correct Charger signature	Read Charger d-/d+ voltage and set correct d-/d+ to match charger and phone current	0 to 5VDC	5V=1.2V to nRF (divide input to 1.2V=2FF)	Probably risk to add this but seems to could work. Must be tested
Uout Level	Send Uout CMD according user requirement setting or Power monitoring	Set T1, T2 and Enable According Table 5.3.1.9	L/H		
Uin/Uout	Request Uin/Uout value	Reply UI/Uout value when requested by App and update display every 10 sec. Turn off charger if Values are outside spec.	0 - 5V	200mV/V 6V=2FF (Res divider 6V to 1.2V)	
Iout	Request Iout	Reply Iout when requested by App and update display every 10 sec.	0 - 3A	1V=3A Vref 1.2V 3.6A=2FF	
Temp	Request Dongle internal Temperature	Send Temp Alarm to App when required. App must turn off charging current and display Alarm message	Use nRF51822 internal temp sensor		
LED – 1-6	Send Battery level to turn LED on corresponding battery charge level from 0 – 100%	Activate LED as defined by App. See table 5.3.1.6	L=Active		
Alarm/Ambient LED	Send Alarm LED color and intensity	FW set LED as defined by App	H=Active		Optional

TABLE VI
CHARGE VOLTAGE CONTROL PROGRAMMING

Charge Current	U out [V]	T1	T2	EN	Comments
Off	0	0	0	0	Turn off Q1 and LDO
Low Charge current	4.6	0	0	1	Select Resistor for 4.6V output
Med Charge current	4.8	0	1	1	Select Resistor for 4.8V output
Max Charge current	5.0	1	0	0	Turn on Q1 for max voltage and disable LDO

mobile application described in the next section. Figure 13 shows the deployment diagram of the cloud application.

The cloud infrastructure offers a reliable way of developing storing and sharing information for different users. It also offers more cost effective, and other benefits such as security or privacy, with little latency drawback. Thus providing the possibility of a personalized battery charging and protection plan. The end user also has little or no knowledge at all about the system behind the scenes but its offered a robust yet easy to use and comprehensive battery protection for their devices.

E. System architecture and communication interfaces

In this section we describe the communication interfaces between the mobile application and the dongle, relying on Bluetooth low energy technology, Figure 14

The GATT (Generic Attribute Profile) service is the core of the communication interface between the hardware dongle and the app. It defines the characteristics and attributes needed

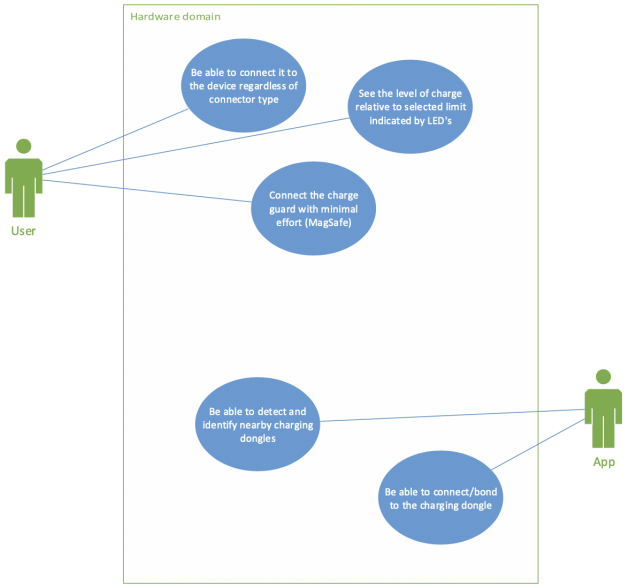


Fig. 11. System usability between the Mobile Application and the User

Fig. 12. Mobile Application UI

in order to transfer data between the phone and the charging dongle. The Universal Unique Identifier (UUID) types in this GATT service table only contains the last 16 bytes of the entire UUID type identifier. The company identifier given by the registration in Bluetooth SIG must be prepended in the application in order to correctly identify the service, characteristic and attributes of this service. The characteristics of the service are:

Level of charge

The level of charge represents the charge level of the phone battery represented as a percentage value between 0 and 100 (integer). The phone gets this value through the APIs presented by the respective operating system. This value is only readable

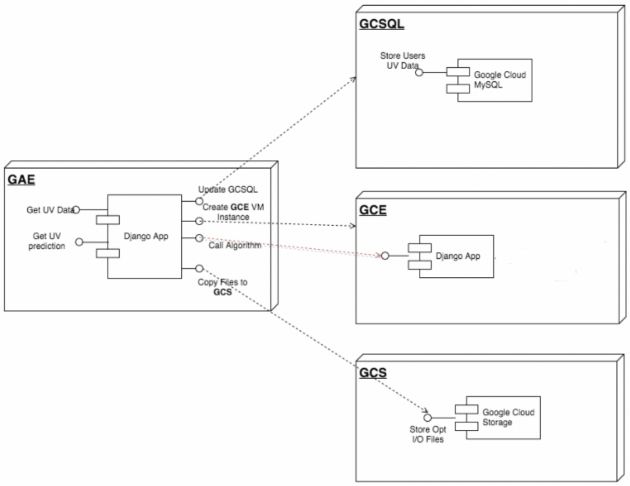


Fig. 13. Google cloud application deployment diagram

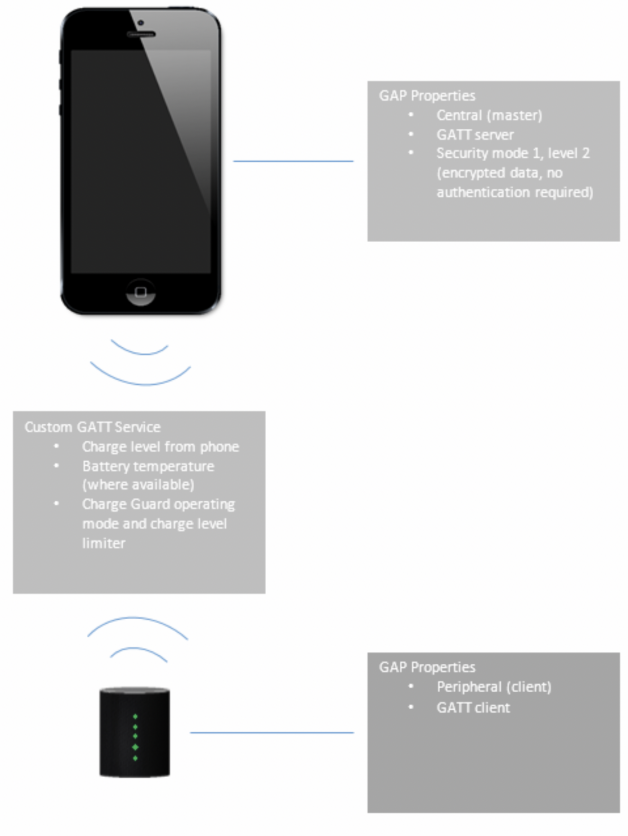


Fig. 14. Communication Interfaces

by the charging dongle (GATT client).

Battery temp

The battery temp represents the current battery temperature of the phone represented as degrees of Celsius scale as an integer. The phones that support this feature will provide the

TABLE VII
GATT SERVICE TABLE

Name (Service 1)	Handle	UUID Type	Permissions	Value	Lenght
Service	0x0090	0x2800 (Primary service)	Read	0x1234	2 bytes
Name (Characteristic1)	**Handle**	**UUID Type**	**Permissions**	**Value**	**Value lenght**
Level of charge (declaration attribute)	0x0091	0x2803	READ	0x0092	2 bytes
Level of charge (value attribute)	0x0092	0x0111	READ|NONE	Actual value as double	8 bytes
Level of charge (descriptor)	0x0093	0x2902	READ WRITE	0x0001| 0x0000	2 bytes
Name (Characteristic 2)	**Handle**	**UUID Type**	**Permissions**	**Value**	**Value lenght**
Battery temp (declaration attribute)	0x0094	0x2803	READ	0x0095	2 bytes
Battery temp (value attribute)	0x0095	0x0112	READ|NONE	Actual value as double	8 bytes
Battery temp (descriptor)	0x0096	0x2902	READ WRITE	0x0001| 0x0000	2 bytes
Name (Characteristic 2)	**Handle**	**UUID Type**	**Permissions**	**Value**	**Value lenght**
Operation Mode (declaration attribute)	0x0097	0x2803	READ	0x0098	2 bytes
Operation Mode (value attribute)	0x0098	0x0113	READ|NONE	Charge Threshold 0-8 byte 9nth byte Bitfield matrix	9 bytes
Operation Mode (descriptor)	0x0099	0x2902	READ WRITE	0x0001| 0x0000	2 bytes

battery temp through the phones API. If the phones API doesnt support retrieval of battery temperature the value should be set to 10 000 so that the charging dongle can identify that the value is not provided and can be overseen.

Operation mode

The operation mode characteristic value is a multipurpose field which represents the following information:

Description is performed MSB:

- Byte 0: Charge threshold (uint8)
- Byte 1: Temp threshold (uint8)
- Byte 2: Current limit (boolean)
- Byte 3-9: Unassigned for possible future use

While the GATT table offering the services of communication among the mobile application and the dongle are presented in Table VII

V. Conclusions

In this paper we have shown the possibility of combining Cloud / Mobile / Wearable devices into IoT grid offering safety benefits to the end users while charging their mobile phones. Battery life prolongation has been discussed. The system can have two operational modes, connected or not with the dongle, but also a well-structured synchronization logic. Main benefits are related to easiness of use, robust, reliable, secure and scalable system, yet with very little latency concerns. However still practical issues remain to be addressed such as device proximity awareness, pairing and handshake protocols, centralized vs. peer to peer (p2p) approach, low latency and real time concerns and integration of devices with very low computational power. A possible continuity could be evaluating the challenges of pairing a large set of devices within small areas.

References

[1] Roy Want, Bill N Schilit, and Scott Jenson. “Enabling the internet of things”. In: *Computer* 1 (2015), pp. 28–35.

[2] Stephen S Yau and Arun Balaji Buduru. “Intelligent planning for developing mobile IoT applications using cloud systems”. In: *2014 IEEE International Conference on Mobile Services*. IEEE. 2014, pp. 55–62.

[3] Hannu Sillanpää et al. “Integration of inkjet and RF SoC technologies to fabricate wireless physiological monitoring system”. In: *Proceedings of the 5th Electronics System-integration Technology Conference (ESTC)*. IEEE. 2014, pp. 1–5.

[4] Muthuraman Thangaraj, Pichaiah Punitha Ponmalar, and Subramanian Anuradha. “Internet of things (iot) enabled smart autonomous hospital management system-a real world health care use case with the technology drivers”. In: *2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*. IEEE. 2015, pp. 1–8.

[5] Geng Yang et al. “A health-IoT platform based on the integration of intelligent packaging, unobtrusive bio-sensor, and intelligent medicine box”. In: *IEEE transactions on industrial informatics* 10.4 (2014), pp. 2180–2191.

[6] *Nordic Semiconductors:* https://www.nordicsemi.com/. 2019.

[7] Nan Chen, Xiaodan Li, and Ralph Deters. “Collaboration & mobile cloud-computing: using CoAP to enable resource-sharing between clouds of mobile devices”. In: *2015 IEEE Conference on Collaboration and Internet Computing (CIC)*. IEEE. 2015, pp. 119–124.