*Workshop Proceedings of the 9th International Conference on Intelligent Environments*
*J.A. Botía and D. Charitos (Eds.)*

*doi:10.3233/978-1-61499-286-8-214*

# A DIY approach to the Internet of Things: A Smart Alarm Clock

Gary SCOTT [a1] and Jeannette CHIN [b]
[a] *Anglia Ruskin University*
*Cambridge, United Kingdom*
*gary.scott@student.anglia.ac.uk*
[b] *Computing and Technology*
*Anglia Ruskin University*
*Cambridge, United Kingdom*
*jeannette.chin@anglia.ac.uk*

**Abstract.** This paper explores how the recent development of low-cost System on a Chip (SoC) boards can be used by the Internet of Things (IoT) DIY community to assist the process of smart object innovation through the example of an intelligent alarm clock. The alarm clock will combine existing traffic and weather web services with local temperature sensor readings to provide a suitable alarm time for the user. Included is a brief review of currently available IoT components and state-of-the-art alarm clocks offering augmented features. Provided is a description and justification of both the hardware and software components of the alarm clock. CPU and memory resource testing demonstrate the computational suitability of the SoC device in the context of an intelligent alarm clock. User feedback regarding the features of the alarm clock provides suggestions for further development.

**Keywords.** Internet of Things, smart alarm clock, Raspberry Pi, System on a Chip, DIY, XBEE, smart objects

## Introduction

Human beings possess strong drivers towards DIY which, aside from economic drivers, include using DIY as an outlet for creativity, taking control, diversification, societal resistance of excessive consumerism and globalisation and the simple desire of wanting to do it for oneself [1]. Recent trends in DIY electronics have provided the tools to enable mass creativity in the Internet of Things (was referred to as embedded-Internet devices back in early days [2]) and Smart Cities / Smart Homes.

Smart objects are a key IoT component and by definition are objects augmented with capabilities such as logic processing, information storage, sensing, real-world event actuation and network communication, made possible by small computers. Smart objects can communicate and interact with each other, their environments, the Internet and human beings [3]. An example of an existing smart object is Karotz [4], an Internet-enabled voice-activated smart rabbit which consumes feeds from a selection of

[1] Corresponding Author.

web services and reads the feeds aloud. Karotz also incorporates an RFID sensor to detect the presence of the RFID tags supplied with it.

This paper focuses on evaluating the use of state-of-the-art SoC technology for the purpose of smart object innovation in the form of a smart alarm clock. The device will be equipped with the ability to combine traffic and weather web service data with local temperature readings and automatically adjust the alarm time for the end user to reach a location at a predetermined time. The immediate envisaged benefits of such a device would be to relieve the burden of journey planning, whilst reducing time spent in traffic congestion and reaching their destination on time.

## 1. Available Technology for Smart Object Innovation

### *1.1. Microcontroller and System on a Chip (SoC) Boards*

The emergence of low-cost and low-powered microcontroller and SoC boards, programmable using high level languages such as C# and Python has contributed towards IoT smart object innovation [5]. Examples of microcontroller boards are notably Arduino (programmed using Wiring), Gadgeteer and Netduino (both programmed using .Net framework). A SoC board is the equivalent of a pc running an operating system. An example of a SoC board is the Raspberry Pi, which runs its own version of Linux OS called Raspbian [6]. Python is the default programming language which is pre-installed in Raspbian, chosen for its ease to program and because it is an *interpreted language,* meaning it can be executed without having to be compiled [6]. A key benefit provided by SoC boards is the ability to run multiple processes on a tiny device suitable for incorporating into smart objects.

### *1.2. Smart Objects and their Interaction with the Real World*

Smart objects need to be capable of both collecting data from their immediate environments and triggering real-world events. This is achieved using electronic sensors and actuators. Today there are a large number of low-cost analogue and digital electronic sensor types available to the DIY enthusiast, including light, colour, flex, force, motion, pressure, temperature, humidity, pulse, accelerometer and tilt [7]. Actuators are available in the form of LEDs, motors, audio output devices and electronic relays to turn devices on and off. Another key feature of a smart object is the ability to transmit and receive data over local networks or the Internet. A number of networking products designed to use low-energy protocols such as 6loWPAN and ZigBee have become popular solutions to creating personal wireless sensor networks. Examples of such solutions are Digi International's XBEE and Ciseco's XRF wireless radios. Their low power consumption makes these radios suitable for transmitting and receiving small packets of sensor and actuation data (unlike standard Wi-Fi radios, which are designed to transfer large multimedia and file data).

## 2. Review of Current State-of-the-Art Alarm Clocks

At the time this paper was written, no commercially available smart alarm clocks with functionality to dynamically adjust alarm times based on weather and traffic conditions had been released. A number of commercial alarm clocks, for example La Crosse WE-8115U-S Atomic Digital Clock [8], are available with augmented features such as indoor / outdoor temperature and humidity readings which are displayed on the clock LCD but no logic is performed with these readings.

### *2.1. Smart Alarm Clock Prototypes*

The Dynamically Programmable Alarm Clock (DPAC), designed by students at Northeastern University in Boston, MA, is a self-setting alarm clock, which uses Google Calendar appointments to set alarm times and automatically adjusts them based on current traffic / weather conditions [9]. Web service requests and alarm time logic are performed by an external web service which feeds the alarm times to the clock.

The Rise alarm clock [10] is another product prototype which monitors and uses traffic conditions to calculate the optimal alarm time. It does not take weather data into account and requires connection to the Internet via a telephone socket.

### *2.2. Smart Phone Alarm Clock Applications*

Smart Alarm Clock Pro ++ is an example of a smart alarm clock application which uses weather conditions to automatically adjust the alarm time. The application also incorporates RSS and weather feeds. The logic determining the alarm time is limited to weather forecasts only and does not take traffic conditions into consideration.

Although downloading a smart alarm application to a smart phone has the benefit of requiring no initial hardware outlay, the phone will require an appropriate stand and power supply to replicate the clock display of a dedicated alarm clock.

## 3. Alarm Clock Features

The key qualifying feature of the alarm clock as a smart object is the inclusion of logical processing of local sensor data and web services to determine the optimal alarm time for the end user to reach their predetermined location at the desired time. The magnitude of the alarm time adjustment is dependent on the severity of traffic incidents and weather forecast. Readings from the local temperature sensor are used to further adjust the alarm time to allow time for motorists to de-ice their vehicles if necessary. The local temperature sensor is included to improve the accuracy of readings specific to the user's local environment. Other features of the smart alarm clock include:

- Scrolling live news headlines sourced from a web service.
- Access to Internet radio feeds.
- Displays current traffic conditions for predetermined route.
- Displays local weather sourced from a web service.
- Hosts its own settings web page, accessible via a computer connected to the same local area network as the alarm clock.

## 4. Architectural Framework of Alarm Clock

### *4.1. Hardware for Alarm Clock*

For the alarm clock to be able to perform the logic for the functions specified in section 3, a SoC board is necessary. The Raspberry Pi has been selected as it is readily available and has an active community of developers sharing projects and technical guidance [11]. As the Raspberry Pi is limited to obtaining its time from the network to which it is connected, the addition of a real time clock (RTC) module ensures the correct time is kept when Internet access is unavailable.

XBEE wireless radios provide communication of the temperature and humidity readings between the local sensor and the Raspberry Pi. The XBEE radios offer low-powered data transmission and a well-documented API [12]. A 20x4 character (HD44780) LCD [13] provides a visual display for the alarm clock data. Text strings longer than 20 characters can be scrolled on the screen. A control pad consisting of five input buttons connected via an 8-bit port expander chip (MCP23008) [14] to the Raspberry Pi's i2c interface provides an input user interface for the alarm clock. A suitable amplification solution and speaker will be connected to the Raspberry Pi's 3.5mm audio out jack to provide audio output for the alarm.

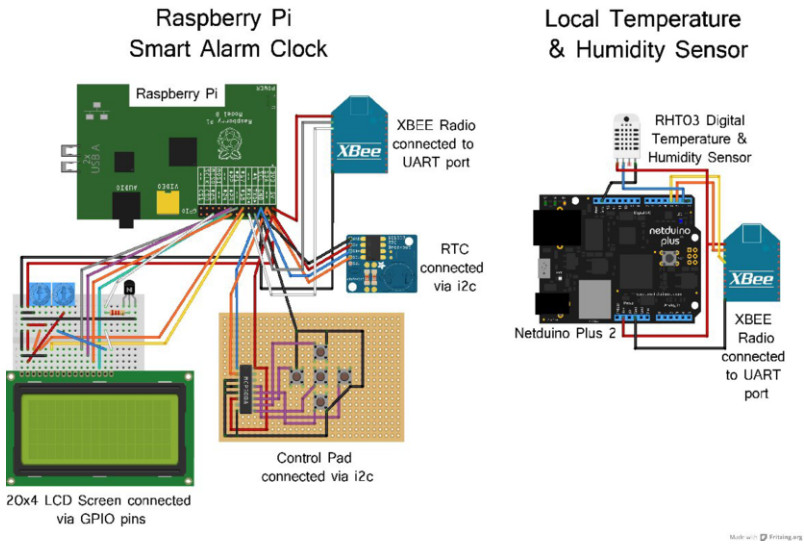


**Figure 1.** Connection Diagrams for Raspberry Pi Smart Alarm Clock and Local Temperature Sensor

### *4.2. Hardware for Local Temperature and Humidity Sensor*

The RHT03 digital temperature and humidity sensor offers high precision and reliable readings [15]. For development purposes, a Netduino Plus 2 [16] microcontroller board has been selected to process the sensor readings and transfer them to the wireless XBEE module via the UART port. To conserve power further, a smaller microcontroller interface could replace the Netduino Plus 2 to enable the sensor to be battery powered.

### *4.3. Software and Web Services*

The alarm clock software will be programmed in Python due to this language being pre-installed on the Raspbian OS along with its pre-existing support for the GPIO pins on the Raspberry Pi. The Netduino Plus 2 is programmed using the .NET framework and will be programmed in C# to collect and transmit the local sensor data.

The traffic web service used for alarm time calculations is Bing Maps REST services [17], which provides a simple RESTful interface for requesting traffic data along a route. Weather data is collected from the DataPoint API provided by the UK Met Office [18] which offers weather data in a simple format. XBEE communication is via the XBEE API to ensure correct transmission of sensor data via checksum error checking. Internet radio is streamed to the Raspberry Pi using the Music Player Daemon (MPD) and accessed via the mpc client [19].

### *4.4. Software Implementation of Alarm Clock*

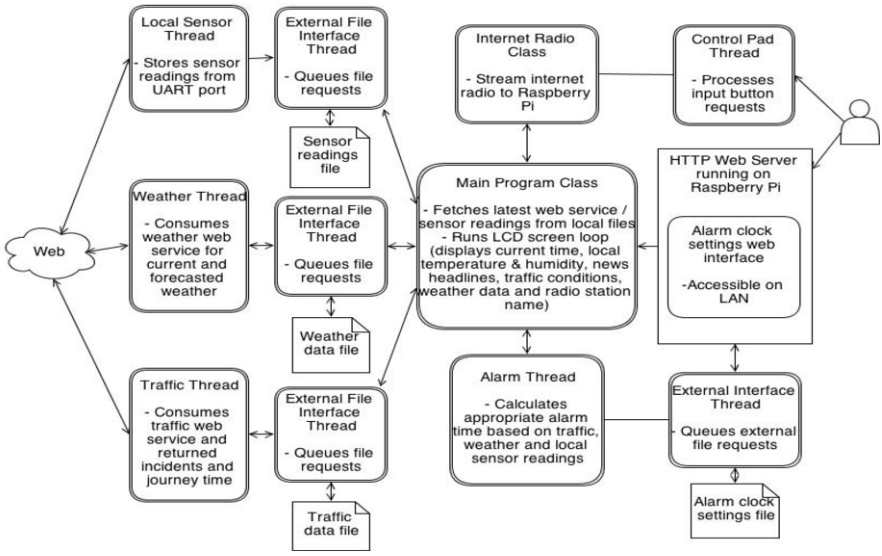


**Figure 2.** Diagram of Software Architectural Framework

## 5. Evaluation

Initial testing has demonstrated that the Raspberry Pi can capably run the smart alarm clock application, concurrently handling the web service and local sensor data requests alongside the LCD output, control pad input requests and its own web server running the settings interface. As reported by Linux's process monitoring facility, 'top', the alarm clock utilises 25-27% CPU and 2.4% memory during standard operation. An additional 12% CPU and 2.5% memory is used whilst Internet radio is active (which spikes to 25% during station changes). This clearly leaves spare processing power and memory available for additional services to be added in future work.

A user trial comprising four IT MSc students at Anglia Ruskin University was set up with the objective of gaining feedback on the features of the alarm clock. Each

student was provided with an overview of the alarm clock before independently evaluating the usability of the LCD module and control pad.

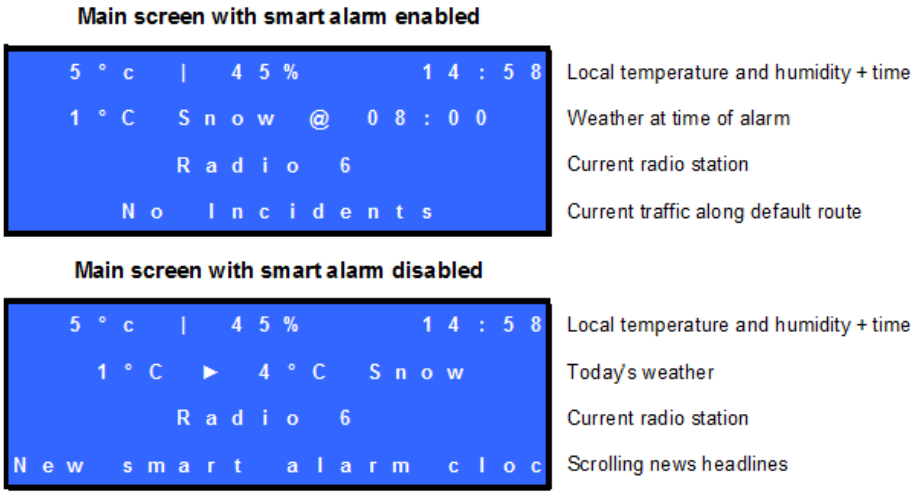


**Figure 3.** Alarm Clock Screen Layout on 20x4 LCD Module

User feedback includes the following suggestions:

- Incorporating a larger screen for greater clarity and addition of new services.
- Interfacing with Google Calendar to display appointments for the day ahead at the time of alarm.
- Providing a more intuitive method of configuring alarm settings via interfaces such as a larger screen, voice commands or a companion smartphone application.
- Porting the application to a smartphone app for use when away from home.
- The addition of social media feeds, for example Facebook and Twitter.

## 6. Conclusion and Future Work

As demonstrated by this work, incorporating SoC boards such as the Raspberry Pi into smart objects, specifically for the purpose of IoT innovation, greatly enhances the local processing capabilities of such objects and negates the usual power consumption overhead of a standard pc. The small physical dimensions of SoC boards enable smart objects to be used in locations where the use of a standard pc would be impractical. As the computational power of the Raspberry Pi is comparable to that of a 300MHz Pentium II PC [20], throughout the design and development phase of the alarm clock, data transfer was limited (where possible) to lightweight standards such a JSON and data storage was limited to local files instead of a dedicated database.

Although the Bing Maps service provides a simple RESTful interface for requesting traffic data along a route, one limitation of using this service is that delay times had to be interpreted from the delay descriptions provided by the service. Google

Maps combines live and historical traffic data to effectively calculate delay time [21] although at the time of writing this paper, a business licence is required to access this information via Google Maps API. The UK Highways Agency publishes its live traffic data in XML format via the Data.gov.uk website [22]. This is a comprehensive source of raw live traffic data but requires translating into geographical routes which is beyond the scope of this project.

In its current implementation, the local temperature and humidity sensor is constrained to transmitting directly to the alarm clock via the XBee wireless modules. Sensor readings could be published to the Internet to services such as Xively (formely known as Cosm and Pachube) [23], which provides an API for other services to consume the readings. When considering a limited number of wireless sensors, this approach could be implemented by transmitting the sensor data directly to the Internet via the alarm clock's WiFi dongle. Whilst using the alarm clock to publish the data would introduce very little hardware resource overhead, adding further sensors may impact upon the performance of the alarm clock. One of the fundamental concepts of pervasive computing is creating environments saturated with sensors and computational abilities, for the benefit of the human occupants [24], thus, as smart objects start to interact with an increasing number of other local network nodes (other smart objects and local sensors/actuators), their computational overhead requirements will also increase. One solution would be to have all nodes transmitting to a dedicated local 'smart' gateway, which can be polled both internally and externally (via the Internet) for local node data. One such gateway that is currently under development is Ciseco's EVA Alpha board for the Raspberry Pi [25], which features multiple wireless sensor network protocol integration onto a single board. A Raspberry Pi then acts as the gateway between local networks and the Internet. The advantages of this model include increased network security via constraining the number of direct connections to the Internet, reduced resource overhead for constrained nodes, a higher level of interoperability between network protocols and the implementation of automated node discovery and subscription [26].

The components used to create the alarm clock are all widely documented by and freely available to the DIY community, enabling mass innovation potential towards the IoT field. By contrast with conventional product design, where the end user is the recipient of a predefined solution, the DIY approach to product innovation is the catalyst towards more personalised and heterogeneous end products [1].

## References

[1] Marc Roelands, Laurence Claeys, Marc Godon, Marjan Geerts, Mohamed Ali Feki, Lieven Trappeniers, Enabling the Masses to Become Creative in Smart Spaces, *Architecting the Internet of Things*, Springer-Verlag, Berlin Heidelberg (2011), 38, 42-43, 40.

[2] Chin J, Callaghan V., Embedded-Internet Devices: A Means of Realising the Pervasive Computing Vision, Proceedings of the IADIS International Conference WWW/Internet 2003, ICWI 2003, Algarve Portugal, 2003

[3] Irena Pletikosa Cvijikj, Florian Michahelles, The Toolkit Approach for End-user Participation in the Internet of Things, *Architecting the Internet of Things*, Springer-Verlag, Berlin Heidelberg (2011), 66.

[4] Karotz: http://store.karotz.com/en_US/

[5] Cuno Pfister, *Getting Started with the Internet of Things*, O'Reilly Media Inc., Sebastopol, CA, USA, 2011

[6] Matt Richardson and Shawn Wallace, *Getting Started with Raspberry Pi*, O'Reilly Media Inc., Sebastopol, CA, USA, 2013

[7] Robert Faludi, *Building Wireless Sensor Networks*, O'Reilly Media Inc., Sebastopol, CA, USA, 2011
[8] La Crosse Technology: http://wwwlacrossetechnology.com/308-145/
[9] Eric Gaertner, DPAC – The Dynamically Programmable Alarm Clock: http://egaertner.com/dpac/
[10] BBC News, Smart alarm clock lets you lie in: http://news.bbc.co.uk/1/hi/technology/2269144.stm
[11] Raspberry Pi Forum: http://www.raspberrypi.org/phpBB3/
[12] Digi International: http://ftp1.digi.com/support/documentation/90000982_B.pdf
[13] Futulec MCP23008 Datasheet: https://www.futurlec.com/SFMicrochip/MCP23008.shtml
[14] HobbyTronics: http://www.hobbytronics.co.uk/lcd-20-4-backlight-blue
[15] Maxdetect: http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Weather/RHT03.pdf
[16] Netduino: http://netduino.com/netduinoplus2/specs.htm
[17] MSDN Bing Maps REST Services: http://msdn.microsoft.com/en-us/library/ff701713.aspx
[18] Met Office DataPoint: http://www.metoffice.gov.uk/datapoint
[19] MPC Linux man page: http://linux.die.net/man/1/mpc
[20] Raspberry Pi FAQs: http://www.raspberrypi.org/faqs
[21] Google Maps: http://support.google.com/maps/bin/answer.py?hl=en&answer=2549020
[22] Data.gov.uk: http://data.gov.uk/dataset/live-traffic-information-from-the-highways-agency-road-network
[23] Xively.com: https://xively.com/
[24] Satyanarayanan, M, Pervasive computing: vision and challenges, *Personal Communications*, *v*ol. 8, Issue 4 (2001), 10-17
[25] Kickstarter, EVE Alpha - Raspberry Pi wireless development hardware: http://www.kickstarter.com/projects/ciseco/eve-alpha-raspberry-pi-wireless-development-hardwa
[26] Tools for the open source Internet of things: http://iot-toolkit.com/