An Industrial IoT Sensor System for high-temperature measurement

Victor Chang and Craig Martin

Artificial Intelligence and Information Systems Research Group, School of Computing, Engineering and Digital Technologies, Teesside University, UK

Email: ic.victor.chang@gmail.com; c.martin0297@gmail.com

Abstract

The world is changing, with more systems becoming automated through the Internet of Things (IoT). By the end of 2020, 38.5 billion connected devices are expected to be part of the IoT, which is an increase of 25.1 billion from 2015. With the digital revolution thoroughly underway, companies have been fully aware that statistical data can help their businesses thrive and increase efficiency. In this paper, the design and implementation of an Arduino sensor system used to record the temperature and location of Ladle Vessels for metallurgical purposes is presented. The Arduino microcontroller has multiple modules connected to it, including a K-Type thermocouple, Global-Positioning System (GPS) Shield, Real-Time clock, and a Bluetooth module. Additionally, our system was fully tested to demonstrate that it could measure 1220 Celsius for twenty-two hours. Our work can contribute to the development of Industrial IoT and smart sensor services.

Keywords: IoT Sensor System; Temperature and Location of Ladle Vessels; Smart Sensor Systems, Industrial IoT (IIoT); Case studies for smart sensors

1. Introduction

The Internet of Things (IoT) is expected to become the latest revolutionary change in the technology industry, affecting all aspects of everyday life. The IoT has rapidly grown in popularity over the past few years, and by the end of 2020, 38.5 billion connected devices could be part of the IoT [9]. Many different industrial sectors are interested in the potential of the IoT and how the collection of data can transform their businesses by automating tasks [1]. The data recorded from the devices can be useful for interested parties to help make important decisions.

For example, IoT sensors are becoming more widely used in the healthcare sector to record patient's blood pressure, sugar levels and weight, with the data being accessible by physicians anytime. This can lead to more lives to be saved due to the constant access to the data [2].

The IoT has several elements: sensors to obtain the data, data processors to analyze the data, and finally, actuators that respond to the information by moving and controlling machinery [1]. David Fletcher states that it is "The ability of objects to communicate that delivers the power of the IoT" in his paper on the Evolution of Cyber Technologies [3]. The communication between IoT devices is not just intended for data acquisition. In many cases, they are meant to perform and complete specific tasks that interact in the physical realm improving efficiency in previously manual tasks [1]. One example is used in the transportation sector, where traffic lights are changed based upon real-time traffic flow information. The traffic lights can take the information from sensors and alter traffic flow based on the data to ease congestion [4]. IoT research and product innovation contribute to the development of Industry 4.0. This paper describes a unique case study that has implemented an Industrial IoT (IIoT) service to offer real-world solutions and applied research contributions.

1.1 Problem Overview

In metallurgy, the temperature of the molten metal entering the casting process affects the quality of the result. Therefore, the temperature of the ladle vessel should be carefully controlled. Ladles are used to transport molten steel to the continuous casting operation, where the liquid is poured into molds to create the casting. The ladles must be preheated to decrease thermal shock and damage to the refractory metals [5]. Once the vessel has reached the optimal temperature, any extra overheating is redundant and can be costly. Material Processing Institute (MPI) is a research and innovation center that develops high-quality bespoke steel castings on a commercial basis. They offer a wide range of steel alloys weighing up to six tonnes. As part of the Industry 4.0 initiative, the aim is to develop and enhance their current IoT solutions to examine data from machinery and equipment further. The company required a live monitoring solution to accurately record and analyze the ladle shell temperature and location during the preheat and transportation phases of the metal making process. The ladle shell has eight unique positions around the edge and on top of the vessel, requiring constant monitoring to accurately assess the current temperature and location status. To successfully record the temperature for long periods without human intervention, the chosen hardware, communication method and programming logic need to be carefully considered to optimize power consumption. This is an industrial IoT (IIoT) project designed, implemented and tested for the MPI in Middlesbrough, UK. Any system developed for MPI would need to be transferable, reusable, and scalable to other locations with minimal effort and adjustments. The network connectivity is weak inside the casting area where MPI develops the metals, so both an offline and online solution is required.

1.3 Proposed Solution

The proposed solution to the problem and the many constraints uses an Arduino microcontroller. Each controller will have an attached thermocouple to gather the current temperature and an external communication module, such as Bluetooth or WiFi, to transmit the data. To investigate the capability of Arduino microcontrollers in a production environment, each Arduino will have some unique modules and functionality. The modules include a GPS shield to obtain the current latitude and longitude of the device, a real-time clock to record the exact moment the temperature reading was taken and an infrared communication module to allow the user to stop the data logging via a remote. The data recorded by the IoT devices will then be displayed through a WPF application (.NET Core, Desktop usage only), with the following capabilities:

1. Display Live Data - Each active device's data displayed on a graph as a live data feed.
2. Display Previously Recorded Data - The user can see all the temperatures and data for each device between specified date ranges.
3. Calculate & Present Statistical Information - Statistical data is calculated for the selected device(s) and

displayed to the user. The statistical data includes the range, mean average and standard deviation of the data set.

The WPF application will be responsible for collecting the data outputted by each nearby Arduino with Bluetooth communication capabilities every second. The received information will then be sent to a local SQL database if the application runs in development mode or to an Azure-hosted SQL database if the application is running in live mode. The Arduino devices that communicate using a network connection send the recorded data to an online Web API, which, in turn, submits the data to the SQL database hosted on Azure. The WPF application can be configured to operate using the local or hosted SQL database with command-line arguments and configuration files. In addition to the WPF application, an Android app will be developed to allow remote access to the data. The Android application will only display previous data to the user and does not include the live or statistical information functionality.

1.4 Background of Problem

Previously, MPI recorded the temperature of the shell using a multi-channel thermocouple data logger, which allows twelve channels of temperature readings to be stored onto an SD card at a user-specified rate. Even though the device was able to successfully and accurately record the temperature data, it had multiple issues and limitations:

1. Costs - Even the cheapest data loggers can be extremely high, which is a problem due to the harsh environment the metal is created.
2. Damage and Replacement Parts - As the device is a specialist piece of equipment, you cannot easily obtain replacement parts, requiring consumers to purchase a new device each time it breaks.
3. Data Acquisition - The device can only log the data to an SD card inserted into the device. Meaning the scalability of such a system is non- existent and requires technologically literate end-users.

4. Unable to Obtain Other Aspects of Real-Time Data - The multi-channel device can only record temperature data and cannot record the current date and time and GPS location of the vessel.

Other similar projects and research outputs have previously been conducted regarding the attainment of temperature data using microcontrollers. This paper structure is as follows. Section 2 describes the methodology for our approach and Section 3 presents the design of the system. Section 4 provides details of the full implementation and Section 5 narrates the testing and evaluation of our work. Finally, Section 6 concludes our paper justifying our contributions.

2 Methodology

2.1 The Waterfall Methodology

The waterfall methodology became the natural fit for the project due to the structure of the task and the different actors involved. The waterfall approach follows a set of steps in a specific order, with the first stage analyzing and designing, followed by the implementation, and finally testing and evaluating the final creation [7]. At the start of the project, a qualitative research interview with MPI was conducted to understand the requirements and expectations of the project before any development was undertaken. A mixture of qualitative and quantitative techniques was well suited for the project as a detailed specification needed to be established before any construction could occur. Overall, the waterfall methodology was a good structure to follow for this project as a clear understanding of the problem was needed before any development could be undertaken. Initially, a specification that later developed into a design proposal was agreed upon by all interested parties. Next, the implementation of the system began and continued until all the agreed features were completed to reach a high standard. The waterfall methodology worked well for this phase of development due to the limited time frame. The waterfall approach gave a structured list of features that needed to be completed for the application to be considered usable and did not allow for new features to be added or continuously changed, which would have increased the development time considerably. Similarly, engineers from Varna University, Bulgaria, developed an Arduino-based device that measured sixteen channels of live temperature data using thermocouples using a waterfall approach. The team used the Arduino Uno microcontroller and K-Type thermocouples to receive the latest readings from electrically resistive furnaces. The objective was to develop a device that could read the latest temperatures and prove the accuracy and precision of the data being recorded [6].

2.2 Information Security

The greatest power of the IoT is the ability for many small objects, such as microcontrollers, to communicate with larger objects, such as desktop computers and servers, to create a network of devices capable of recording data. Security in IoT has to ensure every object in the network can be as strong and protected as possible. If one object was compromised, the entire system could become under threat. The ease of access to IoT devices and connecting them to networks, mobile devices, and other computers without directly requesting permission creates the possibility of high-risk security breaches [8]. With more IoT devices becoming connected than ever before, it is vital that careful security considerations and understanding of the risks are recognized before the deployment of any IoT system [9]. An underlying problem with IoT systems is that they have components that capture environmental data connected to the microcontroller, which can be sent to external servers for storage. For example, the Arduino could record the temperature using the thermocouple and upload the data to a remote server, such as the cloud or a nearby windows device. The issue lies with the non-transparent communication between these devices and the external data servers. These servers can analyze and redistribute the information without the user's direct consent [1]. In 2018, an Amazon Echo recorded a user conversation and then shared it with contacts in her address book without direct consent. The situation was deemed to have been a string of unlikely events after the device woke on a keyword and started listening to the conversation, resulting in an indirect conversation between the device and the owner. The Echo then distributed the recording to all her contacts [10].

With this example, it is important to consider and validate that the data is being sent to the correct locations.

2.3 Industrial IoT approach

Cheng et al. [11] explain an Industrial IoT (IIoT) approach centered on 5G to deliver smart manufacturing services. They explain the architecture. However, the limitation is the lack of the underlying technologies, such as sensors, and how the sensor can cope with memory shortage limitations. Huang et al. [12] explain the use of a blockchain system to enhance security. However, such an approach will lead to high implementation costs, since every system needs to be on the blockchain system. In the current setting and challenge, not every item is required to be in the secure blockchain service. In order to make our work IIoT compliant, operational devices such as sensors can have wireless connections only locally with WiFi and 4G/5G and it requires a strong authentication for accessibility each time. Automation is possible, such as measuring temperature for 24 hours to minimize the level of manual work required.

2.4 Best Practices when Securing IoT Devices

Before undertaking the IoT project for Material Processing Institute, several best IoT security practices have been identified.

1. Physical Protection of Devices - IoT devices should be isolated and protected from unauthorized personnel. If an intruder gained physical access to the device, they could add or remove critical components that affect the usability and security of the device. Furthermore, they could upload different code to distribute the data to alternative servers that are not authorized.

2. Firmware & Patch Updates - IoT devices require hardware to monitor the environment surrounding them. The hardware attached has to be updated and upgraded when required to prevent malicious attacks, which target outdated hardware, from taking place.

3. Fail-Safe Design - As IoT devices connect to other services using some form of communication technology,

there must be a fail-safe strategy in place. For example, if the communication connectivity dropped for an alarm system, a backup must be in place as lives or property could be in danger. For Material Processing Institute, if the Bluetooth connectivity fails temporarily, crucial data could be lost. The loss of data could affect the engineers' decision making during the steel making process and lead to inferior quality steel being produced.

4. Strong Authentication - In order to connect to a Bluetooth receiver, the encryption key must be known. When manufacturers distribute the Bluetooth receivers, they come with default weak passwords such as '1234'. Changing these passwords regularly is good practice as it makes it less likely for attackers to gain access. It is essential to make it a long length with any password, containing an array of lower, upper, and special characters with additional numbers to create a password difficult to crack or guess.

3 Design of the System

When the system is in operation, the IoT devices should be located in isolation boxes attached to the ladle vessel. A nearby windows computer can display the data from each IoT device. If the IoT device uses a Bluetooth module to communicate, the computer will open a network stream and extract it each second.

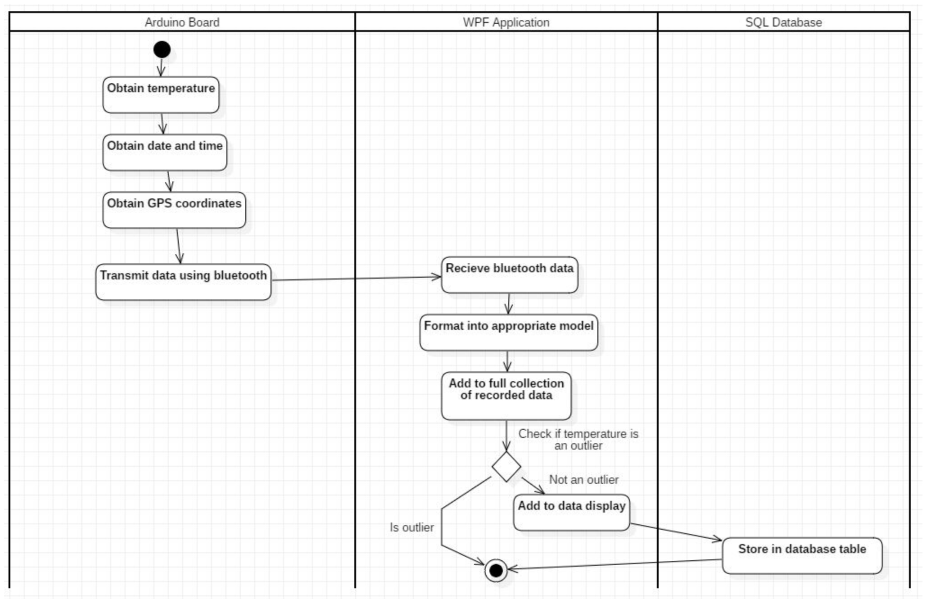


Figure 1: An activity diagram showing a high-level overview of operations between all applications and layers

Otherwise, if the IoT device communicates using WiFi, the data is sent to an online Web API. Applications can then query the API to present the data to users. The WiFi device that has been developed is a prototype for future usage once the network connection has been extended to the ladle vessel area. Therefore, the default method of communication is Bluetooth. Bluetooth was identified as the ideal communication method as it uses less battery power compared to WiFi and has a suitable, although a weaker, range of communication. See Figure 1 for the systemdesign.

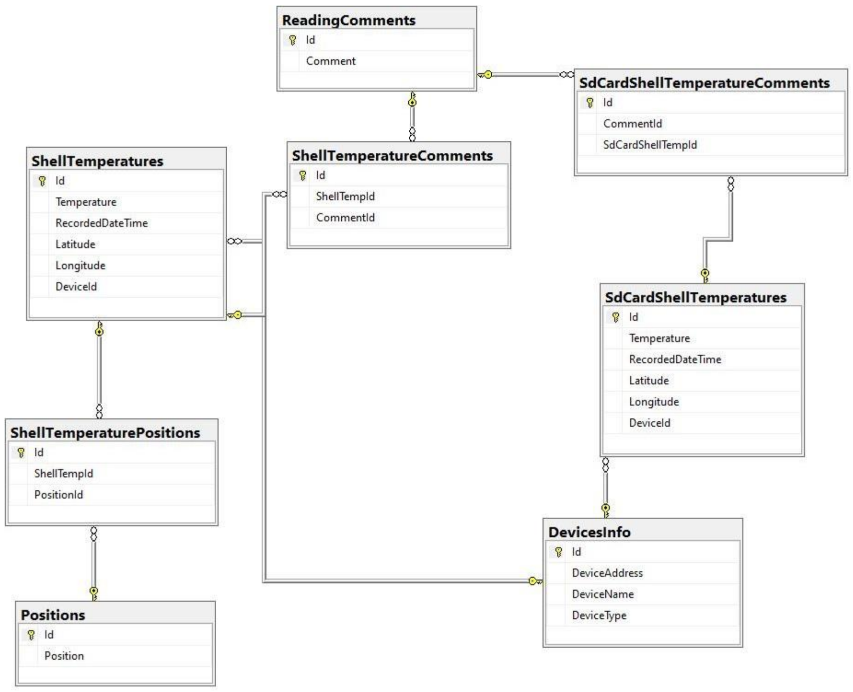


Figure 2: The database diagram with all of the tables and their relationships with one another

Figure 2 shows a database diagram with all of the tables and their relationships, with each component described as follows. Table 1 then shows the required hardware components to build the sensors.

Table 1: Hardware Components

Hardware Components	
Hardware Part	**Description**
Arduino Uno	The Arduino Uno is used for the versions of the IoT device that communicate through Bluetooth. Due to the simplicity of the board and the usage throughout other similar projects.
Arduino WiFi Rev 2	The REV 2 is used to build the IoT device that communicates through WiFi and transmits the code directly to the Web API.
DSD TECH HC-05	The DSD TECH is the chosen Bluetooth module for the project. The Bluetooth module can run on either 3.3V or 5V, has a large range and can detect if it is connected to other devices.

DS3231 Real Time Clock	The real-time clock allows for accurate date and time recording to the latest second.
RoyalTek ITEAD GPS Sheild	The GPS Shield allows for the current latitude and longitude to be recorded.
Type-K Thermocouple	The thermocouple allows for extremely high temperatures to be recorded.
MAX31855K Thermocouple Breakout Board	It allows a connection to be established between the Arduino and thermocouple break- outboard.
Thermocouple Connector PCC-SMP-K	The connector connects the thermocouple to the breakout board.
16GB SD Card	It allows for the storage of data when a connection fails.
LED Lights	To indicate if the device is active or sleeping.
Battery Pack	Power supply to the power board.
KY-022 Infrared IR Sensor Receiver	It receives input requests from the IR remote.
IR Remote	It allows a user to stop the recording of data for the board.

1. **ShellTemperatures** - A table holding information regarding data recordings. Includes the temperature, latitude, longitude, date and time, and the device which recorded the data.
2. **ShellTemperatureComments** - The textual comment associated with the data recording if applicable.
3. **ShellTemperaturePositions** - Stores information regarding which part of the ladle shell the data recording was taken from, i.e., bottom left, top right, etc.
4. **Positions** - A collection of predetermined positions on the ladle shell that users can choose.
5. **ReadingComments** - A collection of previously entered comments for each shell temperature recorded. This table was added to avoid duplication of strings.
6. **SdCardShellTemperatures** - Shell temperature data that was taken from an SD card and not a live recording.
7. **DevicesInfo** - A table to store information for each microcontroller that has previously recorded data.
8. **SdCardShellTemperatureComments** - Comments for SD card recorded data.

3.1 Arduino Design

During the design phase of the project, it was decided that an Arduino would be the best solution due to the plentiful availability and the low cost of the parts. The thermocouple and data communication module (Bluetooth or WiFi) are connected to the Arduino microcontroller and form the IoT device. Additional components are added to different core device variations to evaluate and investigate the IoT's benefits in a steel production environment [13]. The code that is deployed to an Arduino board is called a sketch and is developed in C++. The sketch must include a setup and loop function which provide the following functionality.

- Setup - The setup function is executed at the start of the program and is the entry point of the code. The setup function is intended to set global variables and establish connections [14].
- Loop - The loop function is called directly after the setup function has been completed. The loop function will continuously be recalled by the compiler and the user can insert a delay command to pause the process before repeating it [14].

3.1.1 Base IoT Sensor

A Type-K thermocouple is used for each device in the project because of its ability to record extremely high temperatures and low cost. Additionally, the Type-K has plenty of information online and has been used in other Arduino projects. Other thermocouples were considered, such as Type-J or Type-N, but both are more expensive and are less commonly used [15]. The Type-K probe is inserted into a thermocouple connector, which is soldered onto the thermocouple breakout board. In this case, the breakout board is the MAX31855K, a 14 bit-resolution, SPI-compatible serial interface. It acts as a digitizer, converting an analog reading to digital sends the data out using the SPI interface. The thermocouple breakout board has five pins that must be connected to the Arduino board.

Table 2: Hardware for the Base IoT Sensor

Connecting Breakout Board to Arduino		
Breakout Board Pin	**Arduino Pin**	**Function**
GND	A1	Ground
VCC	A0	Digital pin set to 3.3V (Power)
SCK	D13	Clock
SO	D12	Serial Data Out
CS	D10	Chip Select

The SCK, SO and CS pins are the SPI interface pins and must be connected as specified as the Arduino only has one SPI output. Each IoT device has a connected DS3231 Real Time Clock Module, which allows for the exact date and time the temperature was recorded to be logged. One disadvantage of the module is that the time starts from when the code was uploaded. The module takes the time from the uploading device, stores the date and time and continuously increments both. Once the power is restarted, the time is reset to the original time the code was uploaded and repeats. The Real-Time clock has four pins that must be connected.

Table 3: Connecting the Real-Time Clock and the DSD Tech HC-05 to Arduino

Connecting the Real-Time Clock to Arduino		
Real-Time Clock Pin	**Arduino Pin**	**Function**
GND	GND	Ground
VCC	3.3V	Digital pin set to 3.3V (Power)
SDA	SDA	Serial Clock
SCL	SCL	Serial Data
Connecting the DSD Tech HC-05 to Arduino		
Real-Time Clock Pin	**Arduino Pin**	**Function**
GND	GND	Ground
VCC	5V	Digital pin set to 3.3V or 5V (Power)
Tx	Rx (D0)	Transmitter to Receiver (Comms)
Rx	Tx (D1)	Receiver to Transmitter (Comms)

For the IoT devices that require communication using Bluetooth, the DSD Tech HC-05 is attached to the Arduino board. The module comes with a state pin that can be connected to indicate if the microcontroller is communicating with any other devices, which is essential for storing the data locally when a drop in connection occurs. The Bluetooth module has five pins that must be connected to the Arduino module.

3.1.2 Added GPS Shield to Obtain Latitude & Longitude with Additional SD Card Slot

The Arduino Uno only has one set of SPI pins, which the thermocouple is already connected to (D12-D13). Many GPS and SD Card modules communicate using the SPI interface. One solution would have been to share the SPI interface by changing which SPI device is active on the SPI bus. However, the connection architecture to solve the problem would have drastically extended the life cycle of the project. In order to solve the problem, a GPS shield with an SD Card slot was obtained and connected atop of the Arduino. The thermocouple and real-time clock were then reconnected directly to the shield. The senor and its peripheral hardware have been shown in

Figure 3.

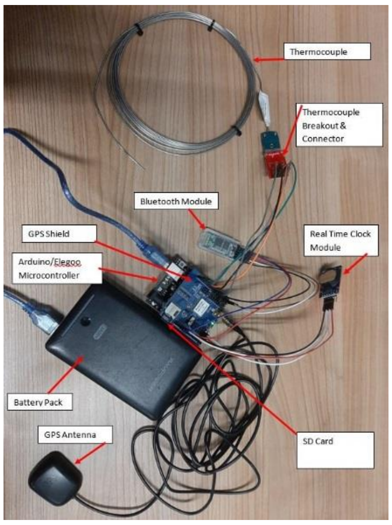


Figure 3: All of the connected modules and hardware required to develop the IoT sensor.

3.2 Desktop Application

In order to allow users to view the recorded data, a Windows Presentation Foundation (WPF) application was developed. Microsoft originally developed WPF for rendering user interfaces in Windows-based applications [16]. The WPF application was chosen over other available technologies such as ASP.NET, AngularJS & React as it can run locally on a computer and does not require a network connection, which is a key requirement of the system. Furthermore, the WPF application could display the information to the end-users to a high and professional standard.

3.2.1 Application Architecture & Structure

The software should be developed to a high quality and maintainable standard, ideally using recent technologies and following modern protocols. After researching online, it was discovered that the recommended architecture structure for any WPF application is the Model-View-ViewModel (MVVM) design pattern. The MVVM pattern separates the programming logic into three different layers [17].

- **Model** – It refers to the encapsulate application data and it can represent a domain model. For example, a model could represent a person with properties, including the person's name, address, age, etc.

- **View** - Views are responsible for defining the structure, appearance and layout of what the user sees on the screen. In WPF applications, the view should be defined in the XAML (eXtensible Application Markup Language), which contains all of the layout code of the view.
- **ViewModel** - ViewModels implement properties and commands that the view can bind to. When the properties are changed, the view can be notified to make the change visually. The ViewModel is the bridge between the View and Model layers and is responsible for handling all interactions between the two.

The WPF application follows an N-Tier architecture pattern that has been built around the MVVM design pattern. Other layers have been included to separate the code further to follow a logical, reusable structure. These additional layers are as follows:

- **Repository** - The repository layer is responsible for communicating with the database. The layer allows the business logic to be decoupled from the data access layer.
- **Data** - The data layer contains all the models needed to communicate with the database. These model classes are Plain Old C# Objects (POCO).
- **BluetoothService** - The BluetoothService is a layer that can search and communicate with nearby Bluetooth devices. The BluetoothService, when reading data from the Bluetooth devices, takes the latest output and looks for certain keys such as "-temp" and extracts the value adjacent to the key, which in this example would be the temperature recorded.
- **Tests** - NUnit tests to validate the correctness of the code.
- **ExcelDataWriter** - The excel data writer takes data collection and writes the data to excel files on the local computer.
- **CustomDialog** - In WPF, the default dialog looks rather old fashioned. The customdialogue was developed to look more professional and is responsible for alerting the users with information and asking for

confirmation of a request before proceeding further.

4 Implementation

4.1 Dependency Injection

A problem with object-orientated applications is that each class can often be dependent upon other modules, which can cause massive problems for developers when making even the smallest of changes. The decoupling of objects is ideal in any software application as it allows developers to replace dependencies without changing classes that use them. A generic interface called ”IRepository” was developed. For each required repository, a real and fake version was created. The real application retrieved data from a real data source and passed it back to the calling function. The fake implementation extracted data from an in-memory collection and returned it to the calling function, developing two different implementations of each repository for thorough testing of the repository layer. If the application was running in development mode, the fake repositories would be added to a service collection and injected into the relevant ViewModels. Alternatively, if the application would be running in live mode, then the real repositories would be injected into the relevant ViewModels. Adding this feature created a more fluid application that was more adaptable and scalable.

4.2 Communicating Data From The Microcontroller to the WPF Application

For users to see the live data that is being recorded, it must be transferred to the WPF application. Firstly, the WPF application searches for any nearby Bluetooth devices with specific names that the user can determine. The user must add the names of the devices to a configuration file to avoid connecting to unnecessary devices and to only connect to relevant microcontrollers and not other objects, such as mobile phones. Once the application has found all nearby devices, a timer is created and started for each one and executes every second. The executed code opens a network stream between the application and the microcontroller and extracts the data from the output stream of the device. The string of data is split into a string array at each new line, and the last element in the array is selected as this is the latest reading. The latest reading is then converted into a string array, separated at each space as to be able to search and extract for specific elements. Each item in the collection (except the last) is checked to see if it matches a key value, such as ”-temp”. Each key value in the string array is used to match a value to a model element. For example, if "-temp" is found, then the next item is the latest temperature reading.

1. ”**-temp**” indicates the next element is the latest temperature.
2. ”**-datetime**” indicates the next element is the date and time the data was recorded.
3. ”**-lat**” indicates the next element is the latitude co-ordinate at which the data was recorded.
4. ”**-long**” indicates the next element is the longitude co-ordinate at which the data was recorded.

Finally, the retrieved data is converted into a POCO object called "LiveDataReading". The object's data is then submitted to the SQL database and the process is repeated.

4.3 Detecting Outliers

While reading the temperature from a Bluetooth device, an occasional anomaly occurred where the readings would drop to zero despite previous recordings being much higher. These results are referred to as outliers. By simply making the temperature integer nullable and ignoring any that were null resolved the problem. Nonetheless, an outlier detection system was needed to accurately filter out any readings that could potentially be anomalies (i.e., considerably less or more than previous readings). The first implementation took the last twenty temperature readings for the specified device and calculated the mean average. Then, if the current reading was five degrees lower or greater than the calculated average, it was considered an outlier and could be ignored. The problem with this solution was it was not fully scientific and allowed for a wide range of error.

After researching online, the interquartile range was identified as the ideal strategy for determining if the latest temperature was an outlier. The new solution took the last twenty temperature readings in order and separated them into three quartiles. Quartile One (Q1) is the median of the first half of the data set, Quartile Two (Q2) is the middle value of the overall set, and Quartile Three (Q3) is the median of the last half of the set. The median of Q1 is subtracted from the median of Q3 to calculate the interquartile range. Finally, the interquartile range is multiplied by a constant value and subtracted from Q1's median and added to Q3's median to generate a lower and upper bound. If the current temperature is less than the lower bound or greater than the upper bound, it is an outlier and ignored. After some trial and error testing, the constant value multiplier was set at 1.75 as it accurately determined outliers. Smaller and larger constant values were tested, but more inaccurate determinations occurred. Overall, the strategy using the interquartile range was much more effective at removing possible outliers as the bounds could dynamically change based upon the data set it was given instead of being hard-coded to specific values—this dynamic change allowed for more accurate detection with more scientific foundations.

4.4 Connection Status Bar

Located at the top of the window is the connection status bar, which indicates the connection status of the current Bluetooth device being recorded. Four main statuses can occur during the application's life cycle.

1. **Connecting** - The background is orange, with a message indicating to wait for the connection to the Bluetooth device to be completed.
2. **Connected** - The background is green with a message indicating that the application has connected to the Bluetooth device and is receiving data.
3. **Failed** - The connection to the Bluetooth device has failed and displays a message accordingly. The application will try to reconnect, although a user may need to intervene if a more critical error has occurred.
4. **Pause & Stopped** - The user has pressed the stop button in the application or has used the IR Remote to stop the Bluetooth device. This triggers the connection status bar to turn grey.

The connection status bar shows the connection status of the currently selected device on the Live Data screen and is viewable from all different (usage of ”but” implies a negative) screens in the application to give the user constant updates.

4.5 Intelligent processing

Between section 4 and section 5.3, the design, system development and implementation have been explained. Intelligent processing can be achieved by following the structured design and development process. In order to make this a highly-efficient process, an additional algorithm for sensors is shown in Table 4.

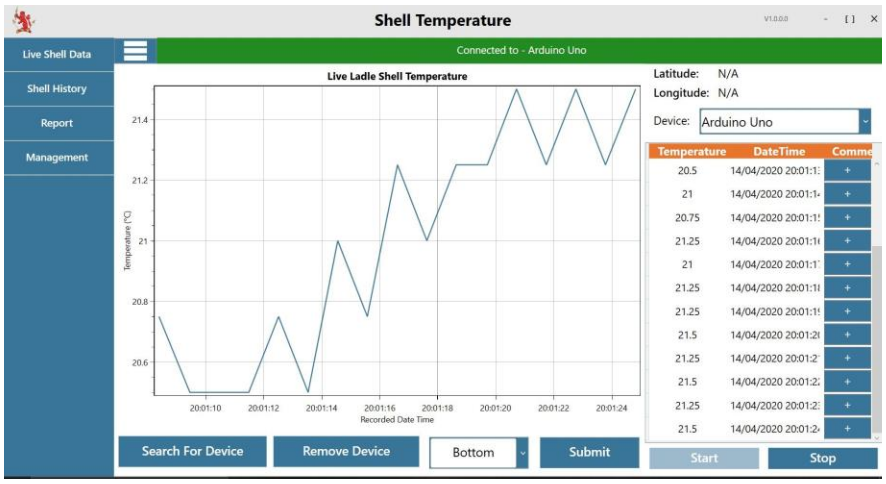
Table 4: An algorithm to run intelligent processing smoothly

```
read( );
record( );
If record(temp) <=1250;
update( );
output( );
end;
else
   alarm( );
output( );
stop( );
exit;
```

We first perform the reading of the temperature and then record the current temperature. If the temperature does not go beyond 1250 C, it will update its current temperature and send the reading to output. Until the sensor reaches its bottleneck, it can trigger the alarm, send everything to output and stop the process. Even we can install additional memory and sensors cannot take too complicated algorithms. This solution effectively ensures our sensors can measure temperature, record and return the outputs until it reaches the bottleneck.

4.6 Displaying Live Data

Once the application has been loaded, the user is presented with the live data screen where the data currently being recorded from each Bluetooth device is displayed. The display consists of a graph that plots each of the temperatures on the Y-Axis against the Date and Time on the X-Axis and a live data feed, which shows all of the recorded data as a list output. The data feed for each device can be viewed by selecting the device from the dropdown menu. Furthermore, the devices' data feed can be stopped and started by pressing the corresponding buttons directly below. The user can add additional information to each record, such as comments or a position tag. Finally, at run time, the user can search for new devices currently not being monitored by the system by pressing the "Search For Devices" button. See Figure 4.



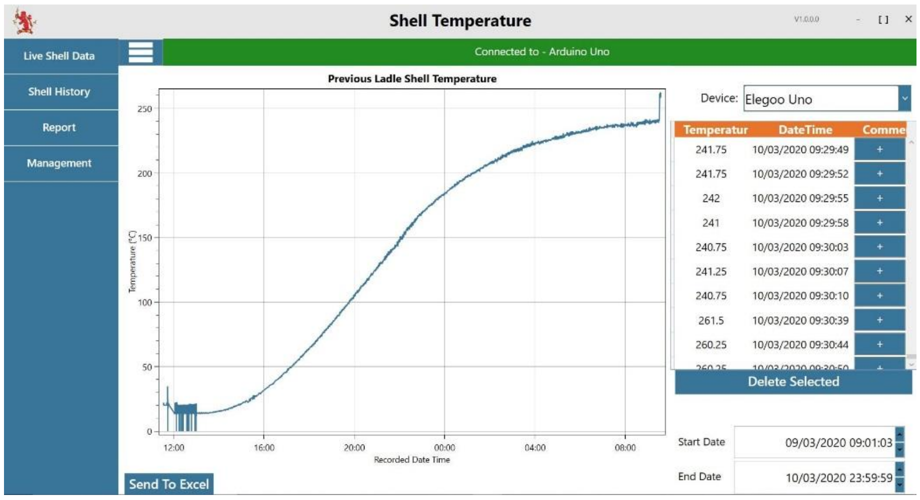
Latitude: N/A
Longitude: N/A
Device: Arduino Uno

Temperature	DateTime	Comme
20.5	14/04/2020 20:01:1	+
21	14/04/2020 20:01:1	+
20.75	14/04/2020 20:01:1	+
21.25	14/04/2020 20:01:1	+
21	14/04/2020 20:01:1	+
21.25	14/04/2020 20:01:1	+
21.25	14/04/2020 20:01:1	+
21.5	14/04/2020 20:01:2	+
21.25	14/04/2020 20:01:2	+
21.5	14/04/2020 20:01:2	+
21.25	14/04/2020 20:01:2	+
21.5	14/04/2020 20:01:2	+

Figure 4: The live data recording screen of the WPF application.

4.7 Displaying Historical Data

Another feature that the user can engage with is the Historical Data view, where the user can view previously recorded data from different devices.



Device: Elegoo Uno

Temperatur	DateTime	Comme
241.75	10/03/2020 09:29:49	+
241.75	10/03/2020 09:29:52	+
242	10/03/2020 09:29:55	+
241	10/03/2020 09:29:58	+
240.75	10/03/2020 09:30:03	+
241.25	10/03/2020 09:30:07	+
240.75	10/03/2020 09:30:10	+
261.5	10/03/2020 09:30:39	+
260.25	10/03/2020 09:30:44	+
260.25	10/03/2020 09:30:50	+

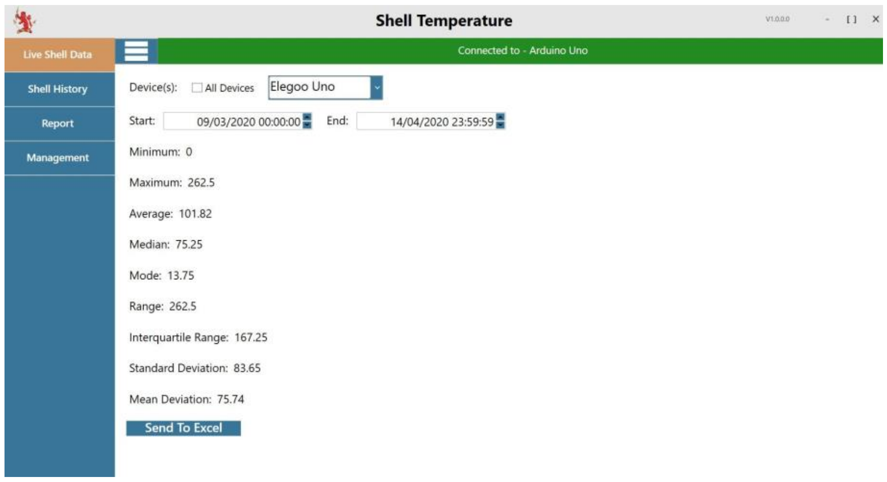
Start Date 09/03/2020 09:01:03
End Date 10/03/2020 23:59:59

Figure 5: The data recorded from one Arduino device during the live trial.

The data is displayed using a graph and data feed. Comments can be added, updated, or deleted against each data recording. The displayed data can be exported to Excel via a button press, shown in Figure 5.

4.8 Calculating Statistical Information

The statistical report view collects all the data readings between a start and end date for a selected device and calculates statistical information based upon the temperature. The statistics include the mean, median, mode, range, interquartile range, mean deviation and standard deviation of the data. The user can then export the data to Excel, shown in Figure 6.



Device(s): ☐ All Devices Elegoo Uno

Start: 09/03/2020 00:00:00 End: 14/04/2020 23:59:59

Minimum: 0

Maximum: 262.5

Average: 101.82

Median: 75.25

Mode: 13.75

Range: 262.5

Interquartile Range: 167.25

Standard Deviation: 83.65

Mean Deviation: 75.74

Figure 6: The report screen where users can see statistical data.

4.9 Web API and Android Application

To allow the Arduino boards to submit the data using a network connection, an endpoint needed to be developed that was hosted on the web. The initial idea was to develop an Azure or AWS IoT service, which is both cloud-hosted platforms used to communicate with IoT devices. However, this was not feasible as synchronizing the data between the existing SQL database and these services would have been challenging and too time-consuming. The settled upon solution was a .NET Core Web API hosted on Azure, which allowed communication to the existing SQL database. The Web API allows the Arduino devices to submit the data readings, and then users can retrieve it by connecting it to a GET endpoint, as shown in Figure 7. The API added a layer of abstraction between the calling application and database and allowed for more reusability in the future.

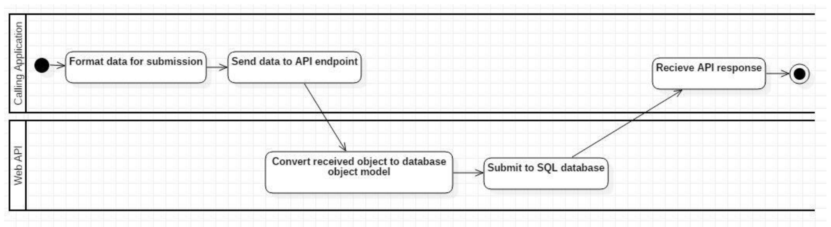

Calling Application
Format data for submission
Send data to API endpoint
Recieve API response
Web API
Convert received object to database object model
Submit to SQL database

Figure 7: An activity diagram explaining the submission process from the calling application, the Web API, and the SQL database.

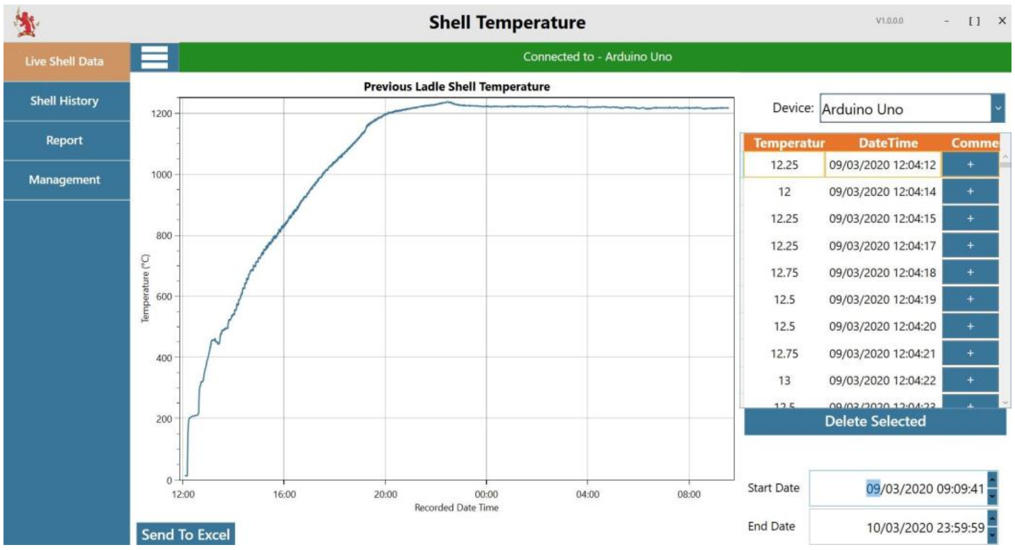
5 Testing and Evaluation of our Solution

5.1 Preliminary & Automated Testing

Throughout the development of the project, extensive manual testing was carried out to check and evaluate that the system was working as expected. With each test, if the functionality could work as expected, the performance of the application was recorded inside a document and signed off from the sprint backlog. To further verify the correctness of the code and automatically check if the code is behaving as expected, 115 NUnit tests were developed. Each unit test defines a set of mock data, passes the data into a function that returns a result, and finally asserts the result against what is expected and will either pass or fail. The functionality that has been evaluated using unit tests includes all calculations, the outlier detector, sorting algorithms, and database interaction, which covers a large percentage of the application's total functionality.

5.2 Live Deployment Testing

On 9 March 2020, at 11:32 AM, the Arduino devices and applications were deployed and successfully recorded the ladle shell temperature for a twenty-two hour period until 10 March 2020 09:32 AM. One Arduino Uno's attached thermocouple was connected to the lower outer edge of the vessel. The other thermocouple was inserted into the top of the vessel. Over the twenty-two hours, both Arduino's began at thirteen degrees and gradually rose until the ladle was fully heated. The lower thermocouple temperature rose to two hundred and two degrees while the top thermocouple rose to 1217 C°. Figure 8 shows the temperature measurement results. We also have a safety measure. When the temperature is above 1250 C, it will stop measuring and trigger the alarm.



Device: Arduino Uno

Temperatur	DateTime	Comme
12.25	09/03/2020 12:04:12	+
12	09/03/2020 12:04:14	+
12.25	09/03/2020 12:04:15	+
12.25	09/03/2020 12:04:17	+
12.75	09/03/2020 12:04:18	+
12.5	09/03/2020 12:04:19	+
12.5	09/03/2020 12:04:20	+
12.75	09/03/2020 12:04:21	+
13	09/03/2020 12:04:22	+
12.5	09/03/2020 12:04:23	+

Start Date 09/03/2020 09:09:41

End Date 10/03/2020 23:59:59

Figure 8: the data recorded from the thermocouple recording the top section of the ladle vessel.

Overall, the live trial was a success as the system managed to record the temperature for the entire period without any connection issues. However, after reviewing the data, one of the devices' early recordings shows that the temperature fluctuated between thirteen degrees and twenty-five degrees for the first hour. It was later realized that old data from the SD card was being transmitted to the application, but it did not have a recorded date and time associated with it. Therefore, the application assigned it to the computer's current date and time and submitted it to the database leading to inaccuracies.

5.3 Evaluation and Discussion

5.3.1 Difficulties and Challenges

One major problem that was overcome during the development process was reading data from the SD card. The difficulty with the task was two-fold. Firstly, the WPF application already had a thread for each device to capture the live data, so it seemed redundant to create a second thread for each device to check if SD card data was then transmitted. Therefore, the SD card data and live data needed to be outputted together. The Arduino library available to read the SD card data only had the inbuilt functionality to read each line individually and could not remove it once read. The second issue was that the Bluetooth connection could drop while reading the data from the SD card. If this event occurred, then the line (position) inside the SD card that had been reached would be forgotten. Once the connection is resumed, the data extraction from the SD card would start at the beginning again, as there is no functionality available to jump to a specific position. The solution was to output the live and SD card data simultaneously at the same moment. In order to overcome the connection dropping during the data reading of the SD card, a loop was added inside the function reading the SD card to write to the text file if the Bluetooth connection failed. Overall, the solution to the problem took a considerable amount of time to achieve and much longer than I anticipated.

5.3.2 Improvements, Future Development & Research

One future improvement in our research is to add a service layer inside the WPF application to communicate with the database. When the application runs in development mode, the service layer would call the repository layer to engage with the database. If the application was running in live mode, the service layer would call it the online Web API to query the database. Adding the service layer would make the code more manageable and allow for further scalability [18]. When Web APIs are deployed online, it is important that they are secure and unauthorized personnel cannot access sensitive data. With the API we have developed, anyone with some programming knowledge could easily query the API as there is no security in place currently. If anyone tries to send or retrieve data from the API, they could do so without any credentials. Without validating the user's identity, a major security issue has been exposed to the general public and attackers. To solve this problem, and IdentityServer should be developed that requires users to register, log in, and pass a token to the endpoint to validate their identity. As aforementioned, the development of similar Arduino recording systems is almost nonexistent, which could be an intriguing subject for future research. It is primarily investigating and developing Bluetooth topology in production environments.

6. Conclusion

Overall, the Arduino system and WPF Application were developed to a high standard and provided all of the functionality required in the specification. The successful delivery of the project is reflected in the live deployment, as the system did not falter during the data logging process. Authors have demonstrated that the IIoT service provides a strong case study of implementing Industry 4.0 disruptive technologies for the Material Processing Institute. Our research contributions come in two aspects. First, the development of the IIoT sensor and smart service. Second, the real-world solutions of the smart sensors to measure the temperature of an oven for a manufacturing firm. It can measure the extremely high temperature periodically and also demonstrate our performance evaluation. Our future work will include improving our sensors to measure the temperature of extreme conditions, such as liquid nitrogen or in regions of high or low temperature, since our sensor can measure up to 1250 C as the highest level and the possible -200C as the lowest level. We will also improve the capabilities of our sensors so that it can be used in other smart services.

Acknowledgments

We pay gratitude to the Material Processing Institute, Middlesbrough, UK, for allowing us to use their facilities and continued support throughout the project. This work is also partly supported by VC Research (VCR 0000055).

References

[1] F. Allhoff and A. Henschke. The internet of things: Foundational ethical issues. Internet of Things, 1:55–66, 2018.

[2] D. Lu and T. Liu. The application of IoT in medical system. In 2011 IEEE International Symposium on IT in Medicine and Education, volume 1, pages 272–275. IEEE, 2011.

[3] D. Fletcher. Internet of things. Evolution of Cyber Technologies and Operations to 2035, page 19, 2015.

[4] R. Mun˜oz, R. Vilalta, N. Yoshikane, R. Casellas, R. Mart' ınez, T. Tsuritani, and I. Morita. of IoT, transport SDN, and edge/cloud computing for dynamic distribution of IoT analytics and efficient use of network resources. Journal of Lightwave Technology, 36(7):1420–1428, 2018.

[5] B. Glaser, M. G¨ornerup, and D. Sichen. Thermal modelling of the ladle preheating process. Steel research international, 82(12):1425–1434, 2011.

[6] K. Yordanov, P. Zlateva, I. Hadzhidimov, and A. Stoyanova. Testing and clearing the high temperature module error from 0 to 1250 c for measurement with 16 k-type thermocouples. In 2018 20th International

Symposium on Electrical Apparatus and Technologies (SIELA), pages 1–4. IEEE, 2018.

[7] S. Balaji and M. S. Murugaiyan. Waterfall vs. v-model vs. agile: A comparative study on sdlc. International Journal of Information Technology and Business Management, 2(1):26–30, 2012.

[8] H. F. Atlam and G. B. Wills. IoT security, privacy, safety and ethics. In Digital Twin Technologies and Smart Cities, pages 123–149. Springer, 2020.

[9] T. Xu, J. B. Wendt, and M. Potkonjak. Security of IoT systems: Design challenges and opportunities. In 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pages 417–423. IEEE, 2014.

[10] N. Chokshi. Is alexa listening? Amazon echo sent out recording of couple's conversation. The New York Times, 25, 2018.

[11] J. Cheng, W., Chen, F., Tao & C. L. Lin. (2018). Industrial IoT in 5G environment towards smart manufacturing. Journal of Industrial Information Integration, 10, 10-19.

[12] J., Huang, L., Kong, G., Chen, M. Y., Wu, X., Liu, & P., Zeng. (2019). Towards secure industrial IoT: Blockchain system with credit-based consensus mechanism. IEEE Transactions on Industrial Informatics, 15(6), 3680-3689.

[13] Y. A. Badamasi. The working principle of an arduino. In 2014 11th international conference on electronics, computer and computation (ICECCO), pages 1–4. IEEE, 2014.

[14] S. A. Arduino. Arduino. Arduino LLC, 2015.

[15] R. Bentley. Irreversible thermoelectric changes in type k and type n thermocouple alloys within nicrosil-sheathed mims cable. Journal of Physics D: Applied Physics, 22(12):1908, 1989.

[16] C. Anderson. Essential Windows Presentation Foundation (WPF). Addison-Wesley Professional, 2007.

[17] J. Smith. Patterns - wpf apps with the model-view-viewmodel design pattern, Aug 2016.

[18] A., Jindal, G. S., Aujla & N., Kumar (2019). SURVIVOR: A blockchain based edge-as-a-service framework for secure energy trading in SDN-enabled vehicle-to-grid environment. Computer Networks, 153, 36-48.

Victor Chang received PhD in Computer Science from University of Southampton, UK. He is currently a Professor in Data Science and Information Systems at Teesside University, Middlesbrough, UK. He is the Conference Chair of 4 international conferences, Associate Editor of IEEE TII and Editor of Information Fusion. He won numerous awards and funding, and is an active and influential researcher.

Craig Martin graduated with First Class Honors BSc in Computing from Teesside University, Middlesbrough, UK. He worked under Prof Chang's supervision.