**RPi-based Website Status Checker with AWS Database Functionality**

By Mikolaj Mroz (2003114)

Internet of Things - CMP408.2022-3.S1.A

Abertay University of Dundee

**Contents**

[Introduction 3](#_Toc125056836)

[Relevance 3](#_Toc125056837)

[Objectives 3](#_Toc125056838)

[Methodology 3](#_Toc125056839)

[Requirements 4](#_Toc125056840)

[System Design (Conceptualisation) 4](#_Toc125056841)

[Development 5](#_Toc125056842)

[Securing 5](#_Toc125056843)

[Conclusion 6](#_Toc125056844)

[References 7](#_Toc125056845)

[Appendices 8](#_Toc125056846)

[Appendix 1 - Text 8](#_Toc125056847)

[Appendix 2 - Screenshots 11](#_Toc125056848)

[References 12](#_Toc125056849)

# Introduction

The aim of this project was to produce a reliable website status checker with Amazon Web Services (AWS) databasing and web hosting functionality on a Raspberry Pi (RPi) Internet of Things (IoT) device. The device used a red and yellow LED to display which of two predefined endpoints (websites) are active, staying lit if there were no issues and beginning to flash if an active ping response was not received. Once the response was processed by the program and the status of the site is determined to be unreachable, an entry was securely submitted to an AWS database with the unreachable address, time of the error, and the error code; this allows for further future investigation into the causes and effects of a potential crash all the while being as easy to use, reliable, and efficient as possible.

## Relevance

Website monitoring is defined as the act of repeatedly checking the status of a website to ensure everything is working as expected and that users can reliably make use of its services and is an important tool in every network engineer’s kit (Queue-it, 2022). The existence of these tools has been documented since the early 1990s with new functionality being added as time passed and technology developed, exemplified by the introduction of real-time network usage graphs, individual user activity tracking (Lord, 2018), and hardware analysis. It was not until the 21st century when these tools began to reach the public eye, with online solutions such as DownDetector and IsItDownRightNow allowing users to freely check the status of their favourite websites and report any crashes themselves, relying on a user-driven model to identify any issues (DownDetector, 2023).

# Objectives

Research

* Secure methods of checking an endpoint’s status
* Uploading data to an AWS database using C MySQL API
* Hosting websites on AWS

Implement

* A ping algorithm that checks the status of two separate endpoints
* An infinite loop to perpetually send out pings reliably
* A function that reads the response received from the ping and determines if the connection was successful
* 2x LED connected to a Raspberry Pi, lighting up on a successful ping and flashing if it failed
* Upon failure, securely upload the time, error code, and the affected address to the connected AWS database.

Methodology

The methodology chosen for this project was the Waterfall Model (Figure 1), which is designed to work well with smaller projects. The model also has a big focus on research without the iterations or repeated supervisor feedback included in models such as AGILE, which are not possible due to the small scale of the project. An important downside to note about the Waterfall model is the difficulty of implementing changes to existing code, which can cause problems for larger scale solutions that require multiple iterations. However, the end goal of the project was decided at an early stage and no major changes were planned, accounted for, or implemented throughout the course of development. The maintenance aspect of the model was not planned to be used within this project, as the “Deployment” stage in this case is the project’s final submission.

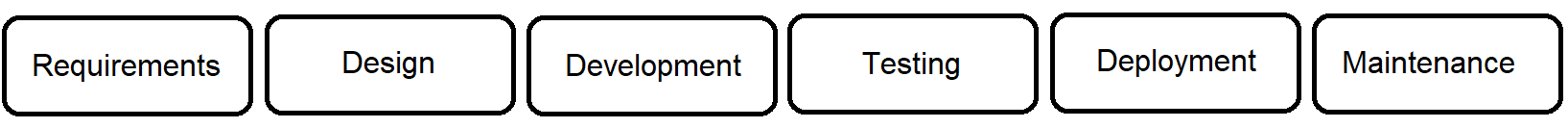


Figure - Waterfall Model (Left to Right)

## Requirements

Firstly, the problem relating to the aims of this project was researched to give a better understanding of the requirements. Results of this research showed that there are currently two main performance monitoring methods: Active monitoring and Passive monitoring (Hein, 2019). Active monitoring requires a script to navigate paths commonly used by either external or internal user requests. Simulating complex user navigation data and requests was not necessary to achieve the aims of the project where a simple ‘ping’ heartbeat implementation sufficed, converting the RPi into a fully-fledged bespoke IoT device. Passive monitoring makes use of real user data to build its results (UpTrends, 2023), though this was not possible due to the small scope of this project, having no active users to draw the data from.

One of the most effective ways of signalling an alert is by creating a visual indicator. This is confirmed by reaction time tests which show that Visual Reaction Time (VRT) sits at approximately 180-260ms across all ages (Figure 2) (Jain, et al., 2015), which allows supervisors of the device to respond in an extremely quick manner to any internal server issues.

Table

Description automatically generated

Figure - A comparison of the reaction time of male and female medical students

## System Design (Conceptualisation)

Considering the results of the research stage of development, several code fragments could be conceptualized for the final program. These were later built off during the development stage with some minor differences, but retaining the core aims of each function. The website and database used by the device were planned to be hosted on AWS Amplify and AWS RDS respectively to meet the requirements of an interconnected cloud-capable IoT device.

See Appendix 1A for a breakdown of the conceptual functions.

The program was conceptualised as a User Application rather than a Loadable Kernel Module (LKM), as it allows for a client to alter the source code to match their desires, scaling it up or down if necessary. Kernel permissions were only to be invoked to make use of keyboard input when writing a password or creating an LED output and are both performed in the background by the program.

## Development

The steps taken over the course of development were largely based on the code fragments identified in the conceptualisation section of the methodology.

See Appendix 1B for real code examples with explanations.

The full code can be found at Appendix 1C.

The C programming language was chosen due its lightweight nature, compatibility with RPi devices, and its extensive standard library. One of the most important library files that was used throughout the project development was the MariaDB/mysql.h which allows a secure connection to a MySQL database.

The database the program relies on was hosted on AWS RDS (Relational Database Service) to meet the requirements of this being a cloud-based device. The database was hosted using default security settings, though it was made publicly accessible to interact with the RPi device. The storage was set to the minimum amount available (20Gb), set to automatically scale with the size of the database in the event it was below or above this amount. To create a fully connected cloud environment, a sample website was hosted on AWS Amplify for the device to ping.

Furthermore, the db.t3.micro instance class was chosen to minimise costs and to avoid using more processing power or storage than necessary. In the case of this project, the database did not have to be powerful or large as it was only ever used by one device, storing very little information in comparison to what it would need to make use of more powerful instances.

See Appendix 2A for a full overview of the AWS database server setup.

## Securing

As man-in-the-middle (MITM) attacks continue to cause problems for network security, it was vital for the program to account for this in the form of encrypted database entries over MySQL to minimise the effects and chances of a successful malicious interception. To add to this, the AWS database storing the error logs collected by the RPi was set to encrypt any stored data (Figure 3). Prepared statements were not necessary as the SQL queries being sent by the program are not customisable by the user without changing the source code, and only cause a big vulnerability risk when dealing with SQL queries determined by the user.

Graphical user interface, application

Description automatically generated

Figure - Storage Encryption page on AWS

# Conclusion

In conclusion, the project meets its aims of being a reliable, effective IoT device with cloud databasing and web hosting functionality. The program was built with security in mind wherever possible and allows for scalability to fit the needs of the consumer, allowing for users to accurately view the status of the defined endpoint addresses and respond accordingly.

Although the program achieves its aims, several difficulties were faced throughout the development stage. Firstly, the initial proposal for the project was written expecting to make use of a small LED matrix to display the errors, however, these were made unavailable shortly after proposal was accepted with the price increasing outside of the scope of the project, and alternatives being out of stock. Because of this, a much simpler dual-LED setup had to be used as a compromise instead. Where the LED matrix would have been easily scalable, displaying the appropriate error on its display and being able to quickly switch between which website is down, the dual-LED setup needs as many LEDs connected as there are websites to check to the detriment of the device’s scalability.

Ideally, the device would also make use of a speaker to sound an alarm, as Auditory Reaction Time (ART) is documented at being much lower across all ages (Kemp, 1973) and making use of both auditory and visual alarms would greatly have improved the accessibility and effectiveness of the device (Focker, et al., 2022).

# References

DownDetector, 2023. *About Us.* [Online]   
Available at: https://downdetector.co.uk/about-us/  
[Accessed 19 January 2023].

Focker, J. et al., 2022. Exploring the effectiveness of auditory, visual, and audio-visual sensory cues in a multiple object tracking environment. *Attention, Perception, & Psychophysics,* 84(1), pp. 1611-1624.

Hein, D., 2019. *Active Monitoring and Passive Monitoring: What’s the Difference?.* [Online]   
Available at: https://solutionsreview.com/network-monitoring/active-monitoring-and-passive-monitoring-whats-the-difference/  
[Accessed 19 January 2023].

Jain, A., Bansal, R., Kumar, A. & Singh, K. D., 2015. A comparative study of visual and auditory reaction times on the basis of gender and physical activity levels of medical first year students. *Int J Appl Basic Med Res.,* 5(2), pp. 124-127.

Kemp, B. J., 1973. *Reaction time of young and elderly subjects in relation to perceptual deprivation and signal-on versus signal-off conditions.,* Washington D.C: American Psychological Association.

Lord, N., 2018. *What is User Activity Monitoring? How It Works, Benefits, Best Practices, and More.* [Online]   
Available at: https://digitalguardian.com/blog/what-user-activity-monitoring-how-it-works-benefits-best-practices-and-more  
[Accessed 19 January 2023].

Queue-it, 2022. *Website crashing? Here’s your recovery plan essentials.* [Online]   
Available at: https://queue-it.com/blog/website-crashing-recovery-plan/  
[Accessed 19 January 2023].

UpTrends, 2023. *What is Synthetic Monitoring?.* [Online]   
Available at: https://www.uptrends.com/what-is/synthetic-monitoring  
[Accessed 19 January 2023].

# Appendices

## Appendix 1 - Text

1A – Pseudocode Concepts

|  |  |
| --- | --- |
| Pseudocode | Description |
| (;;){PING} | A loop that sends out pings to the defined endpoints until the program is closed. |
| Switch checkStatus(AWSwebsite) {  case isDown = true:  blink(LED);  sound(BEEP);  print(“Website is down!”);  uploadToDB(time, errorCode, siteName);  break;  else:  continue;  } | A switch case that determines if the result received from the website is a success or a failure and blinks the LED light accordingly. A switch case was used here because it is easily scalable if more websites were to be added for scanning. |
| uploadToDB(time\_t Time, string ErrorCode, string affectedWebsite){  string username = “username”;  password = userInput(string);  string endpoint = “endpointAddress.com”;  connectSecure(endpoint, username, password);  upload(row1, time);  upload(row2, error);  upload(row3, affectedWebsite);  } | Uploads the necessary information to the AWS database in the event of a ping failure. |

1B – Real Code Examples

|  |  |
| --- | --- |
| Code Examples | Description |
| for (;;){  delay(500);  switch (checkStatus("ping -c1 " WEBSITE1 " -w 2 -qn")){  case 1:  uploadToDB(con, WEBSITE1);  blink(YELLOW);  case 0:  digitalWrite(YELLOW, 1);  } | The function loops indefinitely, checking the status of each website with a ping command and acting on the value returned from checkStatus where 1 = down and 0 = active.  If it is down, the database is updated and the light blinks. If not, the light stays perpetually lit. |
| int checkStatus(char pingCommand[]){  int isDown = 0;  if (system(pingCommand) == 0) {  isDown = 0;  } else {  isDown = 1;  }  return isDown;  } | This function returns isDown = 1 (true) if the website is found to be unresponsive and 0 if it is not. This is later used in the Main switch loop to determine if the LEDs need to blink or not. |
| MYSQL \*con = mysql\_init(NULL);  if (con == NULL){  printf("An error has occured when connecting to the database.\n");  exit(1);  }  printf("Please enter the database password:\n");  char password[32];  scanf("%s", password);  if (mysql\_real\_connect(con, "pingerror-db.cyqhd8liffpt.us-east-1.rds.amazonaws.com", "admin", password, "errorpingdb", 3306, NULL, 0) == NULL)  {  printf("Incorrect Credentials\n");  mysql\_close(con);  exit(1);  } | To connect to an AWS database, a MySQL connector API had to be implemented as shown.  The password must be entered by the user to establish a secure connection. Appropriate error messages are displayed if the credentials or connection encounter an error. |
| void blink(int led){  for (int i=0;i<3;i++){  digitalWrite(led, 1);  delay(PINGTIMING);  digitalWrite(led, 0);  delay(PINGTIMING);  }  } | This is a small function used to blink the chosen LED. The timing of this ping can be determined by the user at the top of the source code, and the number of flashes can be changed by altering the loop value “3”. |

1C – Full code

|  |
| --- |
| /\*  Mikolaj Mroz / Web Status Checker Mini-project  2003114  README:  Program must be run as admin/sudo in order for the GPIO pins to be inistialised.  Otherwise, just use the following in CLI:  sudo gpio -g mode 17 out  sudo gpio -g mode 18 out  \*/  #include <stdio.h>  #include <string.h>  #include <stdlib.h>  #include <wiringPi.h>  #include <time.h>  #include <mariadb/mysql.h> //Use MariaDB  #define MAX\_STRING 128  #define YELLOW 1 //Set Yellow WPiPin (GPIO 18, Pin 12)  #define RED 0 //Set Red WPiPin (GPIO 17, Pin 11)  #define PINGTIMING 100 //Set PING interval (milliseconds)  #define WEBSITE1 "test.d1b6gqf9yd0abh.amplifyapp.com" //Define websites to check  #define WEBSITE2 "gjdfgj.comds" //\*Placeholder unreachable website to demonstrate flash\*  int checkStatus(char pingCommand[]){ //Check ping response, return down status  int isDown = 0;  if (system(pingCommand) == 0) { //If ping returns an error code (code>0)  isDown = 0; //Then return that the website is unreachable.  } else {  isDown = 1;  }  return isDown;  }  void blink(int led){ //Blinks at a defined const interval  for (int i=0;i<3;i++){ //Change '3' to increase or decrease # of flashes  digitalWrite(led, 1);  delay(PINGTIMING);  digitalWrite(led, 0);  delay(PINGTIMING);  }  }  void initialise(){  printf("Initialising Website Status Checker...\n");  wiringPiSetup(); //Initiates wiringPi  system("sudo gpio -g mode 17 out"); //Set GPIO pins to OUTPUT mode  system("sudo gpio -g mode 18 out"); //...Required for LEDs.  } //Change to correct pin numbers if necessary  void uploadToDB(MYSQL \*con, char affectedWebsite[])  {  time\_t t; //Defines time variable  time(&t);  char query[MAX\_STRING] = {0}; //Required to combine variables and strings in snprintf  int errorCode = 1; //returns 1 (isDown value)  snprintf(query, MAX\_STRING, "INSERT INTO pinglogs (affectedWebsite, errorCode, timeCaptured) VALUES ('%s', %d, '%s')", affectedWebsite, errorCode, ctime(&t)); //Craft Query  if (mysql\_query(con, query)) //Submit Query  {  printf("Error submitting Query\n");  mysql\_close(con);  exit(1);  }  }  int main(){  initialise();  //Standard MySQL/MariaDB connection API  MYSQL \*con = mysql\_init(NULL); //MySQL Connector  if (con == NULL){  printf("An error has occured when connecting to the database.\n");  exit(1);  }  printf("Please enter the database password:\n");  char password[32]; //Maximum password is 32 chars long. Change if necessary.  scanf("%s", password); //Passwords should never be hardcoded in  //User must input password themselves  //Define database variables (IP/URL, username, password, database, port)  if (mysql\_real\_connect(con, "pingerror-db.cyqhd8liffpt.us-east-1.rds.amazonaws.com", "admin", password, "errorpingdb", 3306, NULL, 0) == NULL)  {  printf("Incorrect Credentials\n");  mysql\_close(con);  exit(1);  }  //Infinite Loop to repeatedly send out pings  for (;;){  delay(500);  switch (checkStatus("ping -c1 " WEBSITE1 " -w 2 -qn")){  case 1:  uploadToDB(con, WEBSITE1);  blink(YELLOW);    case 0:  digitalWrite(YELLOW, 1);  }  delay(500);  switch (checkStatus("ping -c1 " WEBSITE2 " -w 2 -qn")){  case 1:  blink(RED);  uploadToDB(con, WEBSITE2);  case 0:  digitalWrite(RED, 1);  }  }  mysql\_close(con); //When broken out of the loop (CTRL + C)...  exit(0); //Close MySQL connection and exit  } |

To compile: gcc -Wall -o WebChecker WebChecker.c $(mysql\_config --libs) -lwiringPi

## Appendix 2 - Screenshots

2A – Full AWS database server setup

Graphical user interface, application

Description automatically generated

# References

DownDetector, 2023. *About Us.* [Online]   
Available at: https://downdetector.co.uk/about-us/  
[Accessed 19 January 2023].

Focker, J. et al., 2022. Exploring the effectiveness of auditory, visual, and audio-visual sensory cues in a multiple object tracking environment. *Attention, Perception, & Psychophysics,* 84(1), pp. 1611-1624.

Hein, D., 2019. *Active Monitoring and Passive Monitoring: What’s the Difference?.* [Online]   
Available at: https://solutionsreview.com/network-monitoring/active-monitoring-and-passive-monitoring-whats-the-difference/  
[Accessed 19 January 2023].

Jain, A., Bansal, R., Kumar, A. & Singh, K. D., 2015. A comparative study of visual and auditory reaction times on the basis of gender and physical activity levels of medical first year students. *Int J Appl Basic Med Res.,* 5(2), pp. 124-127.

Kemp, B. J., 1973. *Reaction time of young and elderly subjects in relation to perceptual deprivation and signal-on versus signal-off conditions.,* Washington D.C: American Psychological Association.

Lord, N., 2018. *What is User Activity Monitoring? How It Works, Benefits, Best Practices, and More.* [Online]   
Available at: https://digitalguardian.com/blog/what-user-activity-monitoring-how-it-works-benefits-best-practices-and-more  
[Accessed 19 January 2023].

Queue-it, 2022. *Website crashing? Here’s your recovery plan essentials.* [Online]   
Available at: https://queue-it.com/blog/website-crashing-recovery-plan/  
[Accessed 19 January 2023].

UpTrends, 2023. *What is Synthetic Monitoring?.* [Online]   
Available at: https://www.uptrends.com/what-is/synthetic-monitoring  
[Accessed 19 January 2023].