Arduino fault tolerant System, Internet of Things (IoT)

*[a]Oluwole K. Bowoto, [a]Bankole I. Oladapo, [a]Prince A. Nimako, [a]Francis T. Omigbodun, [b]Omonigho A. Emenuvwe

[a]*School of Engineering and Sustainable Development, De Montfort University, Leicester, UK*

[b]*Department of Mechanica/Mechatronicsl EngineeringAhmadu Bello University , Zaria, Nigeria*

**Corresponding author, Tel: +44(0)7823823066, E-mail: P17239977@my365.dmu.ac.uk*

Abstract

New developments in broadcast technology, such as the ad hoc wireless network and the detection devices that work, enable daily connections to the Internet, commonly referred to as the Internet of Things (IoT). IoT saw this as a support for the development of creative and contextual services and applications. The service can respond to the user's environment and preferences. Finite state machines are recently making waves when the issue of self-controlling systems come into the picture of designs. This research focuses on the design of a fault tolerant system that controls random failures impressed on the network. The method comprises of diodes, Resistors, Optocouplers, ESP8266 WI-FI Module Webpage with button, and two Arduinos which in effect will determine iteratively who to handle a failure at hand modelled as when the button on the webpage is pressed. In this work, a Heartbeat Detection and Monitoring System will be made using Arduino that detect the electrical state using the Pulse Sensor and show the readings in Beats Per Minute on the LED connected a microcontroller. It sends the texts to the server using the Wi-Fi module ESP8266 to signals can be monitored from the monitoring zone via the internet. This is reflected by the declaration of a master state by either of two Arduinos whenever the push button is pressed as observed on the codes. The result obtained from the research shows apparently, the state of the fault tolerance system none power deny of heartbeat condition connected to the LED.

Keyword: ESP8266 WI-FI Module; Signal point of Failure; IoT; Heartbeat Detection

1. INTRODUCTION

The observed so many systems fail for one reason or the other. But it would be noted that most failures leave us with no chance for our usual excuses "Out of our control" as technological development has stretched to it's bound in this century. The degree and causes of most failures when carefully examined, transverses around the following; software reasons, power reasons, design reasons, Carelessness of operators etc. It is still worthy of note that the recent failures which exemplify this subject under discussion and that has motivated this design study to be carried out is the recent Fire disaster that took so many lives in Glenfield [1]. Taking a closer look from a software design point of view, assuming the fridge that caused the large fire had a fault-tolerant system in place, it would have probably shut down the system or sent an instant distressed message via WI-FI (cybersecurity) to the company that will respond accordingly to the situation.

This invention relates to a heart rate measurement device comprising a sensor unit that detects a user's heart rate, a signal processing unit that receives and processes the signal generated from the sensor and a wireless signal transmitting unit that takes the message from the processing unit and the transfer the word out to the configured device. The sensor unit detects the frequency of change of blood density to derive the heart rate, with precision and efficiency of detection of heart rate, cooperating the technique of wireless transmission, and thus our purpose of promoting accuracy of detection and improving the convenience of use is achieved [2]. One proposed design to avert such failure is the adoption Finite state machine. In this work, a Heartbeat Detection and Monitoring System will be made using Arduino that detect the electrical state using the Pulse Sensor and

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 June 2019 doi:10.20944/preprints201906.0034.v1

show the readings in Beats Per Minute on the LED connected to it (Pin 13). It will also send the texts to the server using the Wi-Fi module ESP8266 so that signals can be monitored from anywhere in the world over the internet. It is an excellent source for displaying the data online, and you can access the data from at any time and any place. The system takes the physical input from the pulse sensor by the button on the ESP8266 WIFI, and then the data is processed by Arduino to count the number of pulses and displaying the output. This is reflected by the declaration of a master state by either of two Arduinos whenever the push button is pressed as observed on the codes. The result obtained from the research shows apparently, the state of the Arduino when the button is when power is denied any of the heartbeat and the condition of the LED (output) whenever the button is pressed.

1.1. Apparatus for the design

The device used for this research includes ESP8266 WIFI Arduino module, 2 Arduino Uno microcontrollers, Jumper wires, 1 Breadboard, 4 Diodes, 3 Resistors and 2 Optocouplers. The schematic of the design is shown in fig. 2. Below. The Tx of ESP8266 is connected to the Rx of The Arduino board, and the ESP8266 Rx are connected to the Arduino Tx. The ESP8266 is in. The CH_PD pin is connected to 3.3V with a resistance of 10K ohms. The state of the Arduino data is sent to a server using the ESP8266. This data can be now accessed can be edited by the user from anywhere. Table 3 shows the workflow of our step model. The digital pin 8 of both Arduinos are connected to D1 of the Esp8266 module. D3 and D0 are connected via a voltage divider to pin 7 of both Arduino to monitor the heartbeat of the Arduino to ascertain the state of the at any given time. Analogue pin A0 on the Esp8266 module is connected to the led pin via a voltage regulator for monitoring of the heart beats on the serial monitor. Here the bits representing the voltage on the monitor tends to be high when the LED is faulty or disconnected and slightly low when the LED is in good condition. In my design, it is about 834bits when it is in good condition and 1024bits when it is faulty. Lastly, both grounds of the Arduino and that of the Esp8266 module are connected to establish a common ground.

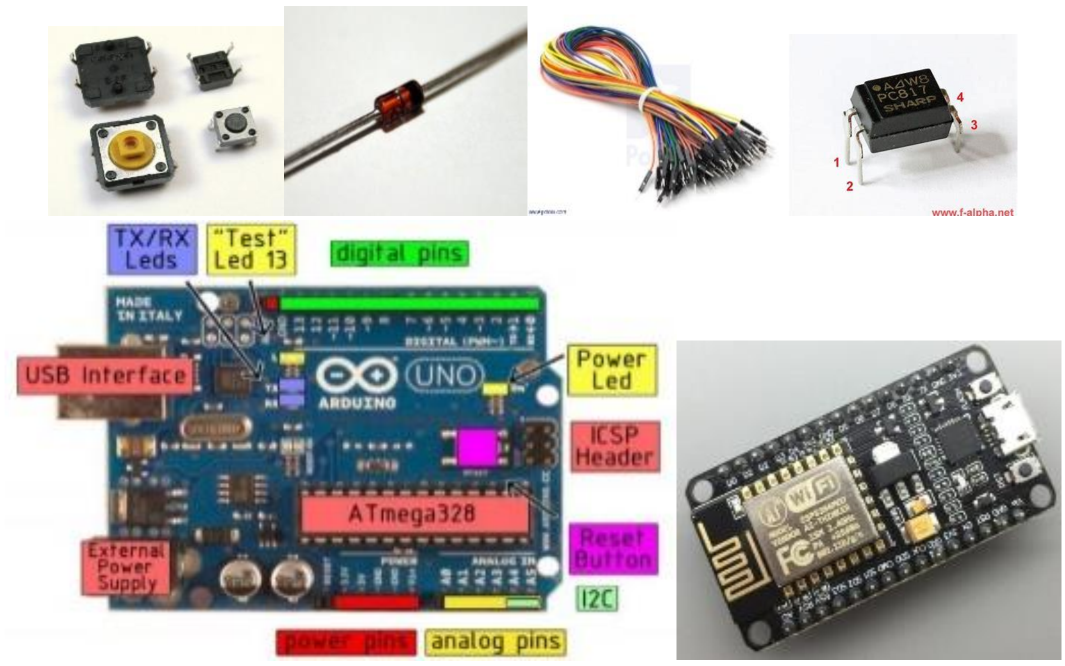


Fig. 1 Component for the system

2. Finite State Machines

A Finite state machine as hinted in the last design study is a device that navigates through some sets of a predefined sequence of states in a predetermined manner.it is a mathematical abstraction of reality that follows a sequential logic to design systems algorithm. In simpler terms, the machine reads a series of inputs and switch to a different state. In this case, every state specifies the next state for every given input. They are deficient level and are usually used to model simple decision-making processes and systems. The operation of a state machine is twofold: It moves through an ordered sequential state in which the next state is defined by the disposition of the following land, depending on the input condition and current state and It produces outputs based on a transition of the rules. The outputs are generated by relying on the present state and shape of the input to the system.

Finite state machines (FSM) and their variants have been widely used as a fundamental semantic model for several behaviour specification languages. Several test case generation techniques have been developed for hardware and software testing based on FSMs; A general description of these techniques is given in [3-5]. All FSM-based test techniques require basic test templates to meet specific essential validation criteria, such as connectivity and minimality. Recent attempts [5] have been made to extend FSM-based test techniques to SPL, mainly using the delta-oriented approach for SPL modelling. In the publication of this document by the conference [5], we proposed the finite state machines (FSM) with a focus on the essential criteria for the validation of the test model for SPL at the family level. However, the problem of scalability is the main problem of using the FFSM model for large and sophisticated systems. Finding a problem makes expensive model analysis to run and leads to a test model that is difficult to maintain. To improve scalability in modelling, we propose this extension to High-Level Hierarchical State Machines (HFSM) called FFSM. The HFSM model is based on state graphs [6], and its syntax is validated by checking well-formed states and transitions. To define a formal semantics for HFSM and allow its legal analysis, we define a transformation from HFSM to FFSM. A general model of the state machine consists of two other essential elements apart from the inputs and outputs: combinatorial logic and memory (registers). The records can be viewed as the functional blocks that make up the system, while the memory acts as a storage of the different states between transitions [12].

2.1. Finite State Machine Variables of Interest

The input which can be any form of stimulus from the outside world, Often a switch or binary sensor. This affects the state which is described as the current condition of the state machine, and there is a limited amount of rules for a given state machine determined basically by its history. This will provide an output know as the linked to finite state machine states but are not the same, and they depend either on the state only or both state and inputs [14]. An advanced machine basically can be considered a black box, and the inputs and outputs can be defined and modified control variables. Due to many data elements, a great way to identify them in groups shows groups of groups of inputs and outputs [15]. Input signals go to the machine different refresh rates and are synchronised and taken at the same speed before entering the state device. A state only works in the state machine, and the system goes to the next state when the transfer conditions are met. The output of the status device is the value of the assets, loads, and control signal. Control input signal, device status output signal makes prices [16, 17]. For the sake of clarity, the input and output signals are defined by their full name. For example, "Planning Islanding" refers to a planned archiving message from ESP8266 [18]. The state machine should work aa fixed frequency/sampling rate of several milliseconds in seconds. The periodic pulse signal can be used as a start signal to start the state device with a time limit.

2.2. Functions Performed by Finite State Machine.

The following are some of the functions a state machine design carries out during operation to enable parallelism and concurrency in executing different tasks which include; Arbitration, Event monitoring, Multiple condition testing, Timing delays, Control signal generation 19]. In this study, the aim is to design a self-validating system or controller out of range that will be fault tolerant or single point of failure (SPOF) free. In effect, the order will be one that receives failure but handles it accordingly and strategically without external intruder. In this project, an Arduino based system will be created, in which the hardware includes an ESP8266 Wi-Fi Arduino module, two Arduinos, one LED, wires, breadboard, resistors, optocouplers and diodes. This Arduino based system will work as follows when the button on the web page is pressed the LED must light up if at least one Arduino is powered up and the ESP8266 wi-fi Arduino module will simultaneously monitor the state of the Arduino, whether it is still powered or not. Precisely one of the Arduinos must control the LED at any time. In this study, the aim is to design a self-validating system or controller that will be fault tolerant with no SPOF (single point of failure) and the controller will be inspected if it is still active. In effect, the system will be one that receives failure but handles it accordingly and strategically with an external intruder. The ESP8266 is an integrated chip adapted to the needs of the Internet. It offers a completely independent Wi-Fi network with which it can act as host or applications all functions of the Wi-Fi network. This module is installed on an Arduino microcontroller board for Wi-Fi. Functionality [20-21]. Arduino reads the pulse every minute Internal A/D and calculates the heart rate, then update to a server connected to the same network.

2.3. Design calculation

Voltage Divider Calculation to Step down Arduino 5v to 3.3v obtainable from the Esp8266 module.

$$V_{out} = V_{in} \frac{R_2}{(R_1 + R_2)}$$

$$3.3 = 5\frac{R_2}{(R_1 + R_2)}$$

$$3.3(R_1 + R_2) = 5R_2$$

$$3.3R_1 = (5 - 3.3)\, R_2$$

$$\frac{R_1}{R_2} = \frac{1.7}{3.3} = 0.52$$

In this design study, taking $R_1 = 1k\Omega$

Therefore $R_2 = \frac{R_1}{0.52} = \frac{1}{0.52} = 1.92k\Omega$

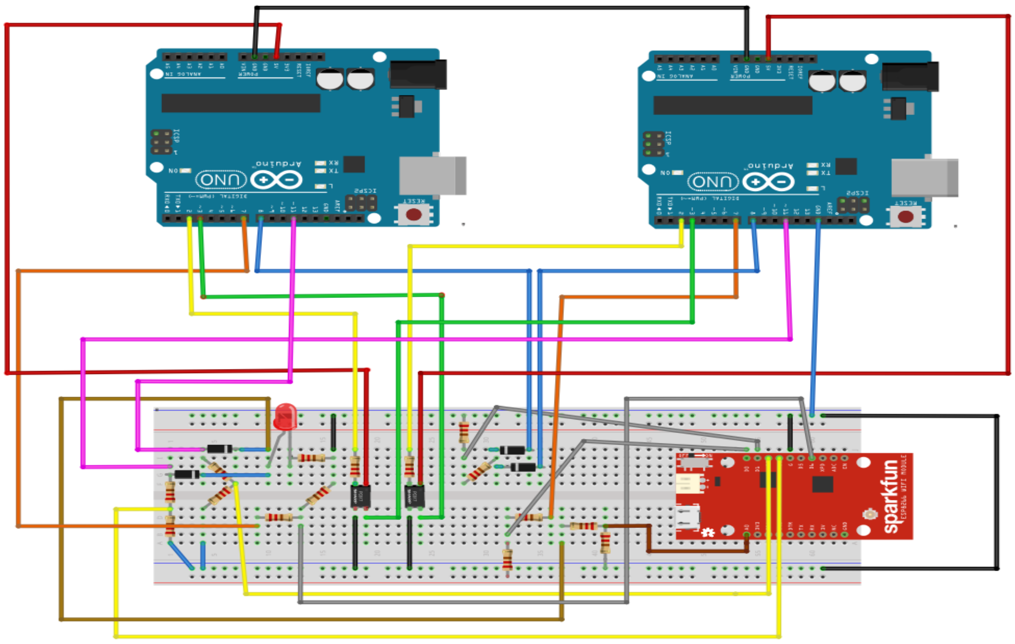


Fig. 2. Circuit Diagram of Design

3. Essential semantics of this system

The Esp8266 module does the following in this aspect of the design, which effectively simulates the button. It must monitor the voltage on the LED to be able to ascertain if the LED is “LOW” or “HIGH”. The ESP also follows impulse (Heartbeat) from the Arduino to determine whether the Arduino is powered or not. In this system, the ESP is the single point of failure; if it fails, the entire system is dead.

3.1. Working principle

Visual Paradigm is a software development tool to implement the proposed machine in the state of the micro networks. That is it can convert graphics presentations, including status transition diagrams, flow charts, transition tables and application management logic for different applications. Example, process sequence, synchronisation

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 June 2019 doi:10.20944/preprints201906.0034.v1

and fault management corresponding to events, fault conditions and other external input signals. The Figures 3a, 3b, 3c and 3d below shows the inside view of the state machine highlighting the operation in the different components. Each algorithm Developed in these figures is implemented in the system state and is defined as Function. The States, as shown for the separate state machines, has two integrated functions: one for regular operation and one for an emergency operation.

3.2. The finite state machine of Arduino module

In this system, the Arduino runs three finite state machines to ensure efficient communication between the ESP8266 Wi-Fi Module and itself. The finite state machines include, the physical signal from the web page also needs to be denounced because it might be noisy sometimes. The Arduino function as the heartbeat, which gives out pulses in the form of voltage to the ESP8266 Wi-Fi Module. In the Wi-Fi module side of things, this heartbeat finite state machine is used to monitor the led voltage at any given time to ascertain whether it is “on”, “off”, or “power outage”.

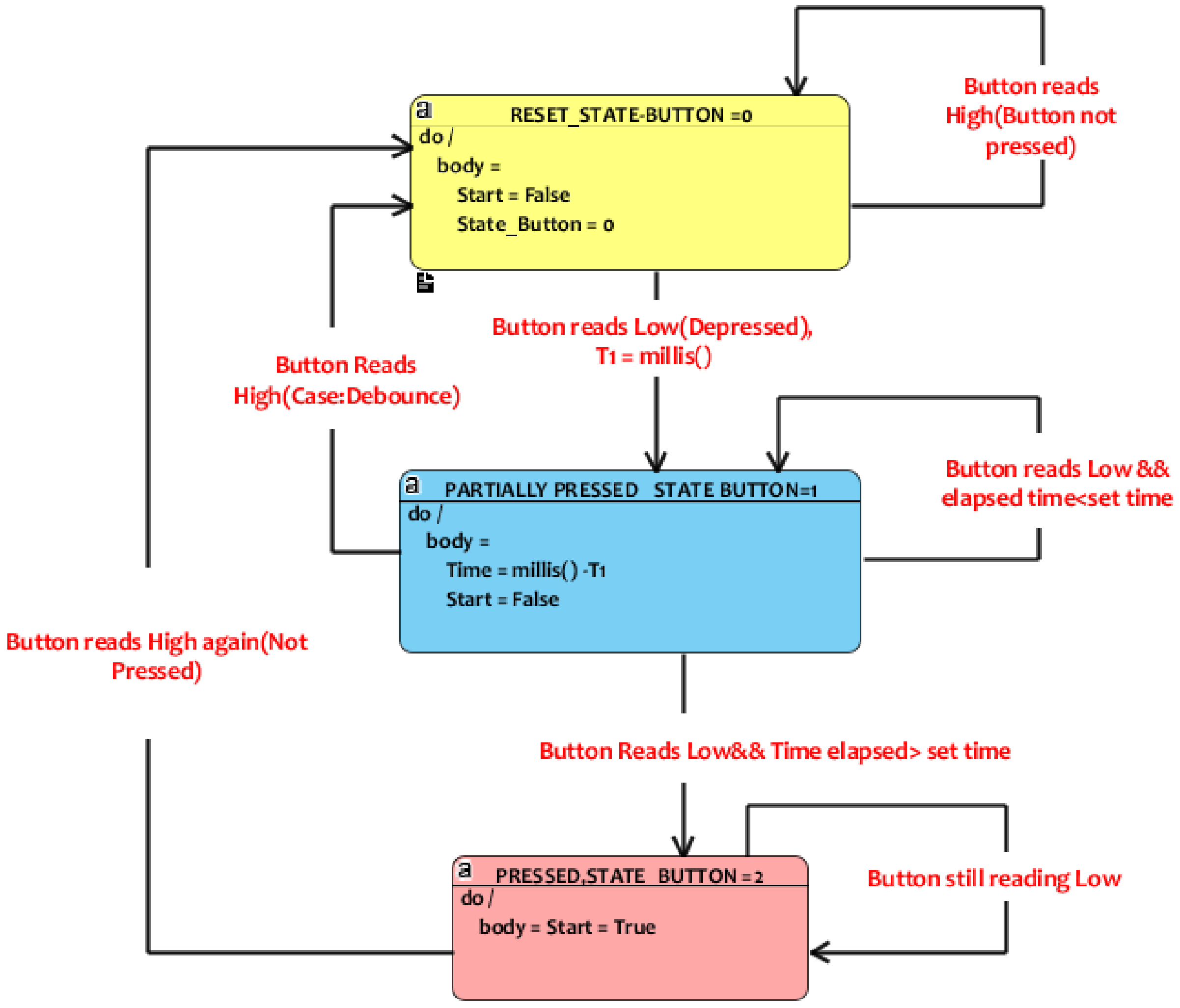
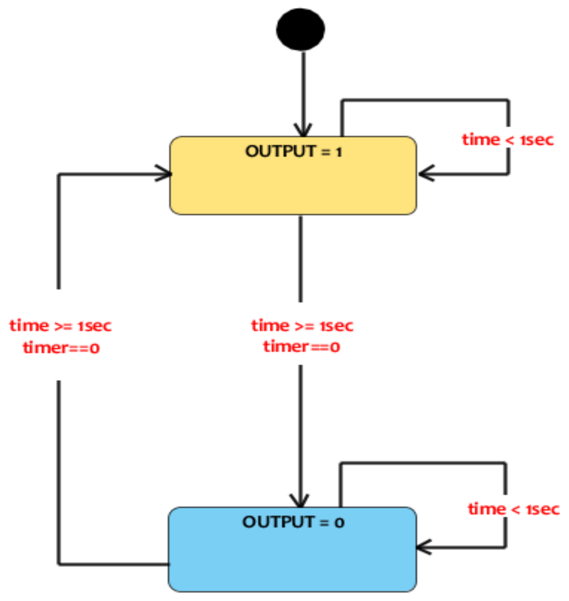

RESET_STATE-BUTTON =0
do /
body =
Start = False
State_Button = 0
Button reads High(Button not pressed)
Button reads Low(Depressed), T1 = millis()
Button Reads High(Case:Debounce)
PARTIALLY PRESSED STATE BUTTON=1
do /
body =
Time = millis() -T1
Start = False
Button reads Low && elapsed time<set time
Button reads High again(Not Pressed)
Button Reads Low&& Time elapsed> set time
PRESSED,STATE BUTTON =2
do /
body = Start = True
Button still reading Low

Fig. 3a. Finite state machine debouncer for the button on the web page

During the bridge failure time of Collision state and the state, the machine must transfer the algorithm to the predefined reactive power support algorithm in state one which is the idle state. All the states each have a built-in function in control references depicted in the Arduino coding environment. This is especially important for the transition algorithm since it should be the same between the current state and the previous one to ensure smooth transitions. The exceptions are the emergency modes, which have predefined action sequences. The transition function is applied according to the following elements. When the output of the logic operation is "high"/"1", the device switches from the current state to the next state. All logic signals like ‘’Reset’’ constitute the input of the

doi:10.20944/preprints201906.0034.v1

state machine. Because each path transition logic completes other paths, only one track can be enabled, and there is no transition conflict. The emergency mode is not intended to be a priority for safe use for standby. However, when the emergency logic becomes "important", the system automatically switches to a predetermined algorithm in case of emergency.



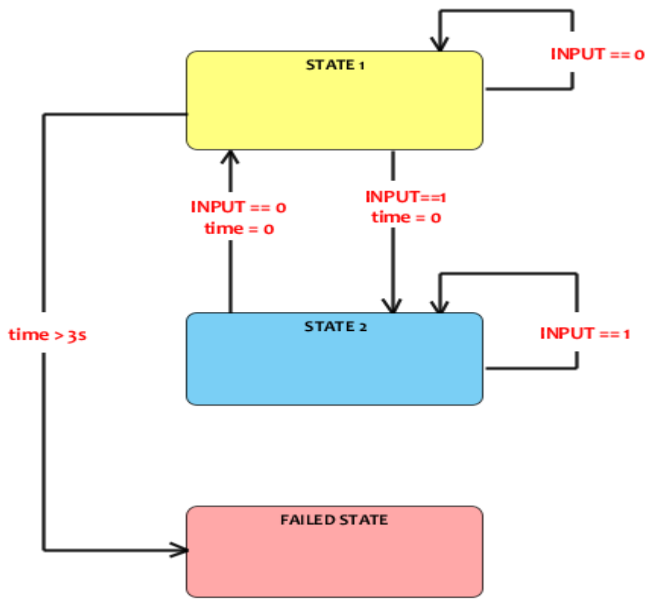


Fig. 3b Heartbeat of Arduino (c) LED Voltage state of Heartbeat of ESP8266

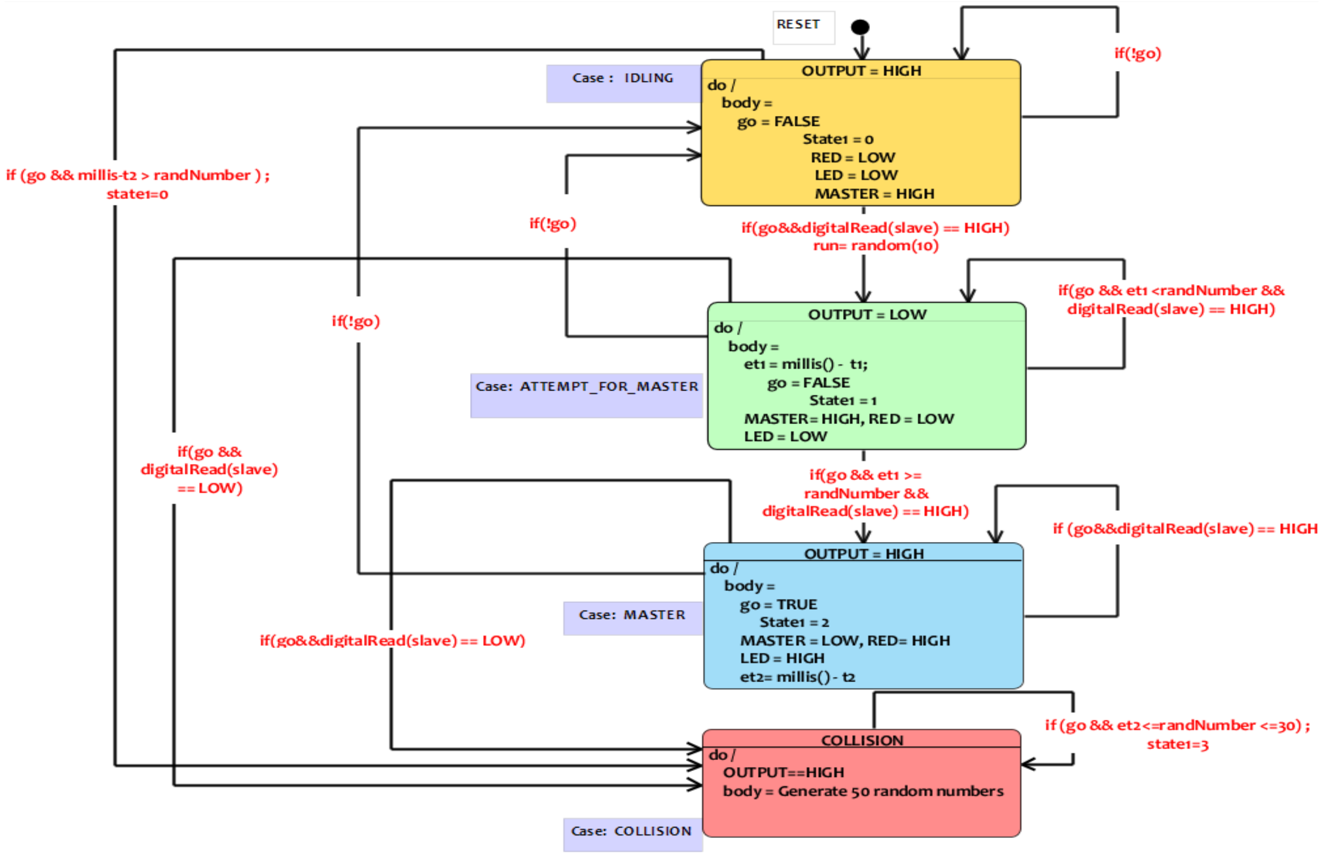
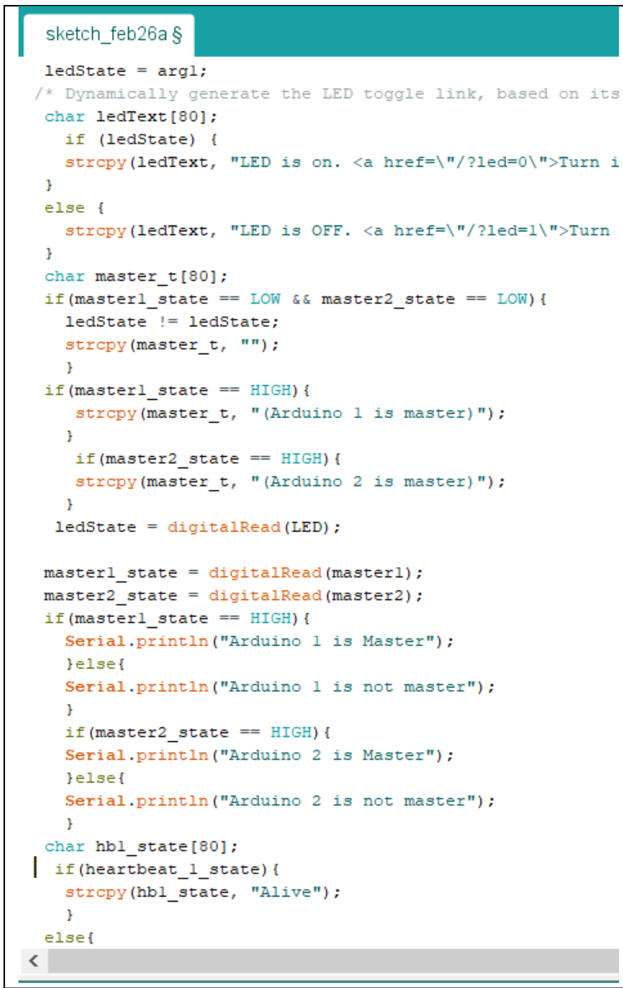


Fig. 3d. Master-slave finite state machine

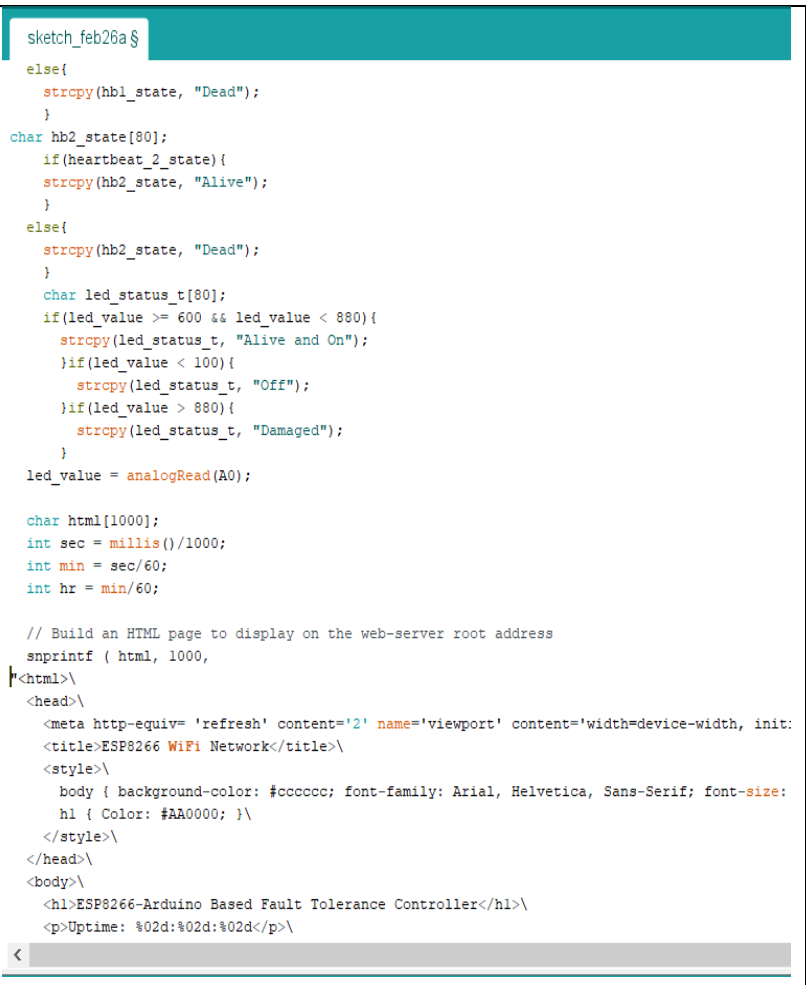
3.3. Esp2866 FSM transition

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 June 2019 doi:10.20944/preprints201906.0034.v1

At the transitions, as shown in the schematics above, the system will figure out a random number. Hence a timer is set up to aid a check and criteria for navigation in between the different states. In the first state, the inputs to the system will be monitored. If the information = 0 or LOW, it stays in the stay in the state and controls the timer set. In any case, if the timer is more significant than a set threshold time, then the system goes into a failed state. But if the input is equal to 1, it moves to the next state as depicted in fig.2 and remains in this state as long as input equal to 1. In this system, the ESP8266 WIFI MODULE runs two finite state machines to establish stable communication and fault detection between itself and the Arduino. The finite state machines include;

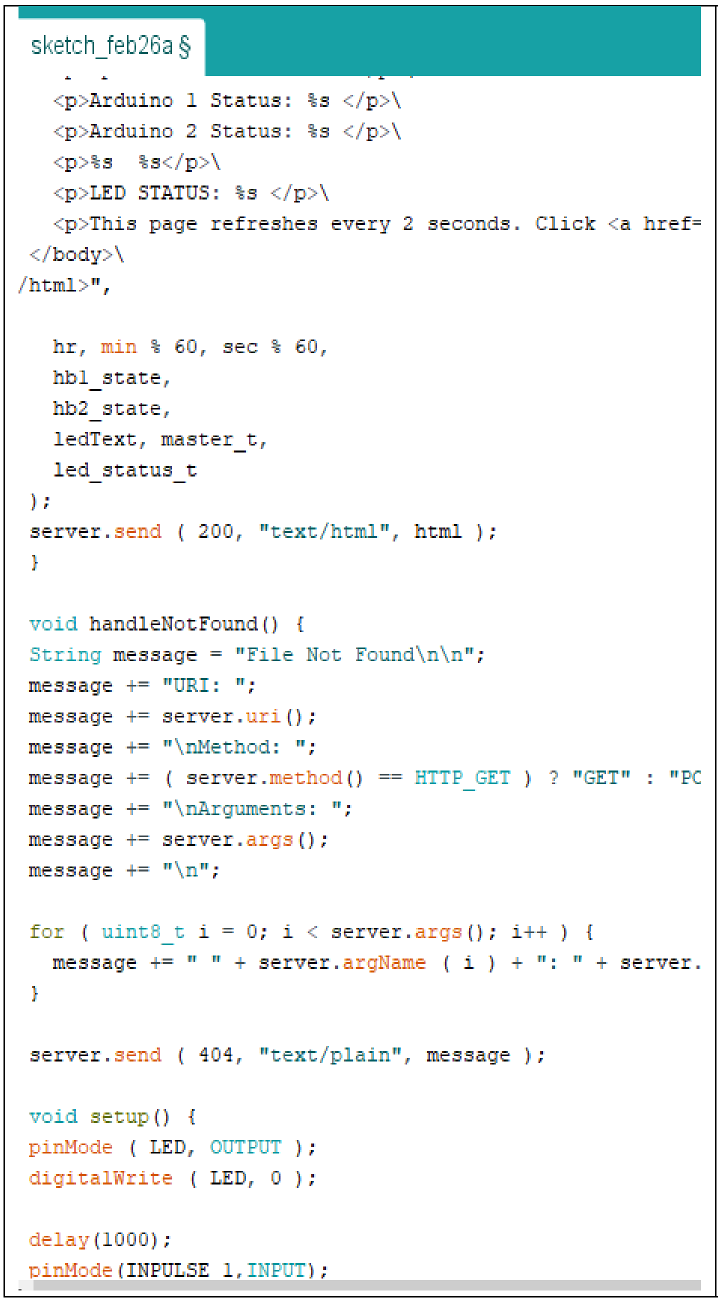


```
 ledState = argl;
/* Dynamically generate the LED toggle link, based on its
 char ledText[80];
   if (ledState) {
   strcpy(ledText, "LED is on. <a href=\"/?led=0\">Turn i
 }
 else {
   strcpy(ledText, "LED is OFF. <a href=\"/?led=1\">Turn
 }
 char master_t[80];
 if(master1_state == LOW && master2_state == LOW){
   ledState != ledState;
   strcpy(master_t, "");
   }
 if(master1_state == HIGH){
    strcpy(master_t, "(Arduino 1 is master)");
   }
    if(master2_state == HIGH){
    strcpy(master_t, "(Arduino 2 is master)");
   }
  ledState = digitalRead(LED);

 master1_state = digitalRead(master1);
 master2_state = digitalRead(master2);
 if(master1_state == HIGH){
   Serial.println("Arduino 1 is Master");
   }else{
   Serial.println("Arduino 1 is not master");
   }
   if(master2_state == HIGH){
   Serial.println("Arduino 2 is Master");
   }else{
   Serial.println("Arduino 2 is not master");
   }
 char hb1_state[80];
  if(heartbeat_1_state){
   strcpy(hb1_state, "Alive");
   }
 else{
```

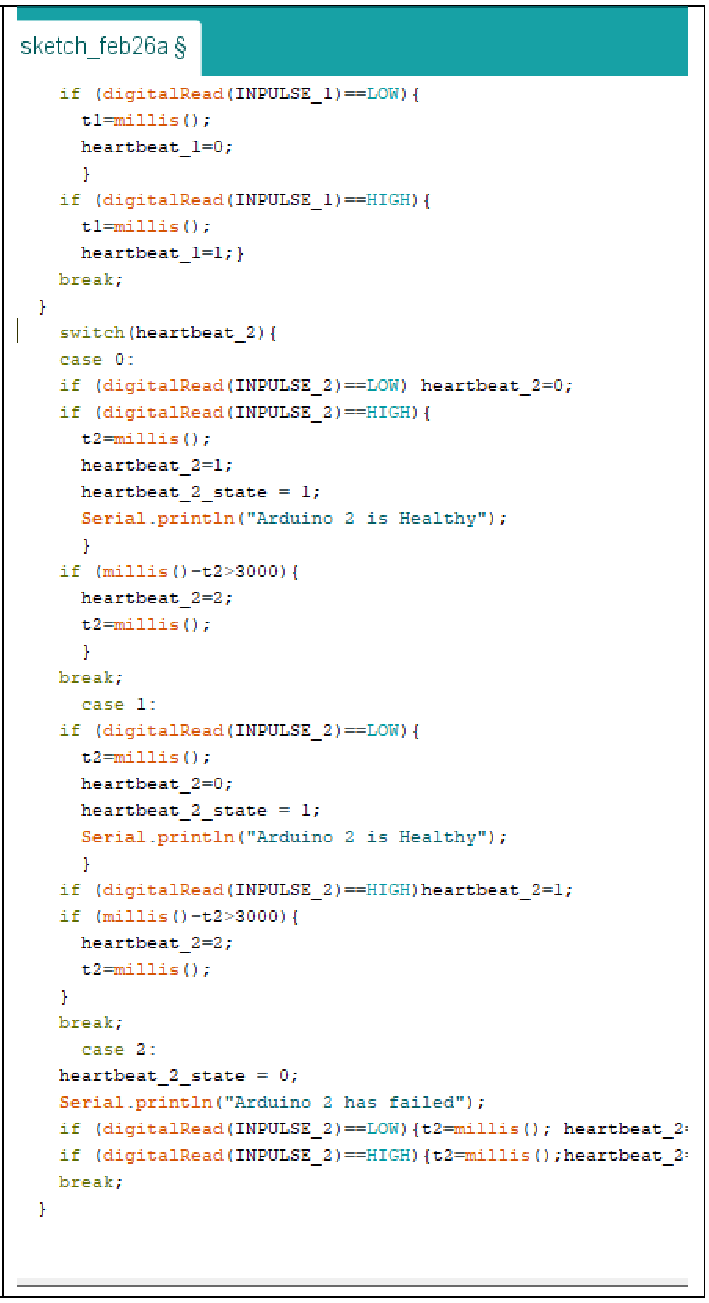


```
  else{
    strcpy(hb1_state, "Dead");
    }
char hb2_state[80];
    if(heartbeat_2_state){
    strcpy(hb2_state, "Alive");
    }
  else{
    strcpy(hb2_state, "Dead");
    }
    char led_status_t[80];
    if(led_value >= 600 && led_value < 880){
      strcpy(led_status_t, "Alive and On");
      }if(led_value < 100){
        strcpy(led_status_t, "Off");
      }if(led_value > 880){
        strcpy(led_status_t, "Damaged");
      }
  led_value = analogRead(A0);

  char html[1000];
  int sec = millis()/1000;
  int min = sec/60;
  int hr = min/60;

  // Build an HTML page to display on the web-server root address
  snprintf ( html, 1000,
"<html>\
  <head>\
    <meta http-equiv= 'refresh' content='2' name='viewport' content='width=device-width, init:
    <title>ESP8266 WiFi Network</title>\
    <style>\
      body { background-color: #cccccc; font-family: Arial, Helvetica, Sans-Serif; font-size:
      h1 { Color: #AA0000; }\
    </style>\
  </head>\
  <body>\
    <h1>ESP8266-Arduino Based Fault Tolerance Controller</h1>\
    <p>Uptime: %02d:%02d:%02d</p>\
```

Fig. 4 (a) Code for the web page (b) Arduino code for master-slave initiation, button and LED



```
    <p>Arduino 1 Status: %s </p>\
    <p>Arduino 2 Status: %s </p>\
    <p>%s  %s</p>\
    <p>LED STATUS: %s </p>\
    <p>This page refreshes every 2 seconds. Click <a href=
  </body>\
/html>",

    hr, min % 60, sec % 60,
    hb1_state,
    hb2_state,
    ledText, master_t,
    led_status_t
  );
  server.send ( 200, "text/html", html );
  }

  void handleNotFound() {
  String message = "File Not Found\n\n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
  message += ( server.method() == HTTP_GET ) ? "GET" : "PC
  message += "\nArguments: ";
  message += server.args();
  message += "\n";

  for ( uint8_t i = 0; i < server.args(); i++ ) {
    message += " " + server.argName ( i ) + ": " + server.
  }

  server.send ( 404, "text/plain", message );

  void setup() {
  pinMode ( LED, OUTPUT );
  digitalWrite ( LED, 0 );

  delay(1000);
  pinMode(INPULSE_1,INPUT);
```



```
    if (digitalRead(INPULSE_1)==LOW){
      t1=millis();
      heartbeat_1=0;
      }
    if (digitalRead(INPULSE_1)==HIGH){
      t1=millis();
      heartbeat_1=1;}
    break;
  }
    switch(heartbeat_2){
    case 0:
    if (digitalRead(INPULSE_2)==LOW) heartbeat_2=0;
    if (digitalRead(INPULSE_2)==HIGH){
      t2=millis();
      heartbeat_2=1;
      heartbeat_2_state = 1;
      Serial.println("Arduino 2 is Healthy");
      }
    if (millis()-t2>3000){
      heartbeat_2=2;
      t2=millis();
      }
    break;
      case 1:
    if (digitalRead(INPULSE_2)==LOW){
      t2=millis();
      heartbeat_2=0;
      heartbeat_2_state = 1;
      Serial.println("Arduino 2 is Healthy");
      }
    if (digitalRead(INPULSE_2)==HIGH)heartbeat_2=1;
    if (millis()-t2>3000){
      heartbeat_2=2;
      t2=millis();
    }
    break;
      case 2:
    heartbeat_2_state = 0;
    Serial.println("Arduino 2 has failed");
    if (digitalRead(INPULSE_2)==LOW){t2=millis(); heartbeat_2=
    if (digitalRead(INPULSE_2)==HIGH){t2=millis();heartbeat_2=
    break;
  }
```

Fig. 4 (c) Arduino code for the button (d) Arduino codes for the LED

4. Result and discussion

When the button on the webpage is pressed, the led on the breadboard comes on (HIGH) and one of the Arduino pin 13 light goes on indicating that it is the master (i.e. the one in control at that moment.).The button on the page is pressed again, one of the Arduino pin 13 light comes on(HIGH) after a random decision made between both Arduinos via the code written above. Also, when the power of one Arduino is removed (Failure condition model), remain powered via the optocoupler and diode in the system always to keep the Arduino always powered. At this point, the second Arduino might take the position of master or the Arduino which lacks power from a source might retain masterhood. This characteristic exemplifies the fault tolerant nature is described as a master must be arrived at no matter the condition of the system at any point in time.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 June 2019 doi:10.20944/preprints201906.0034.v1

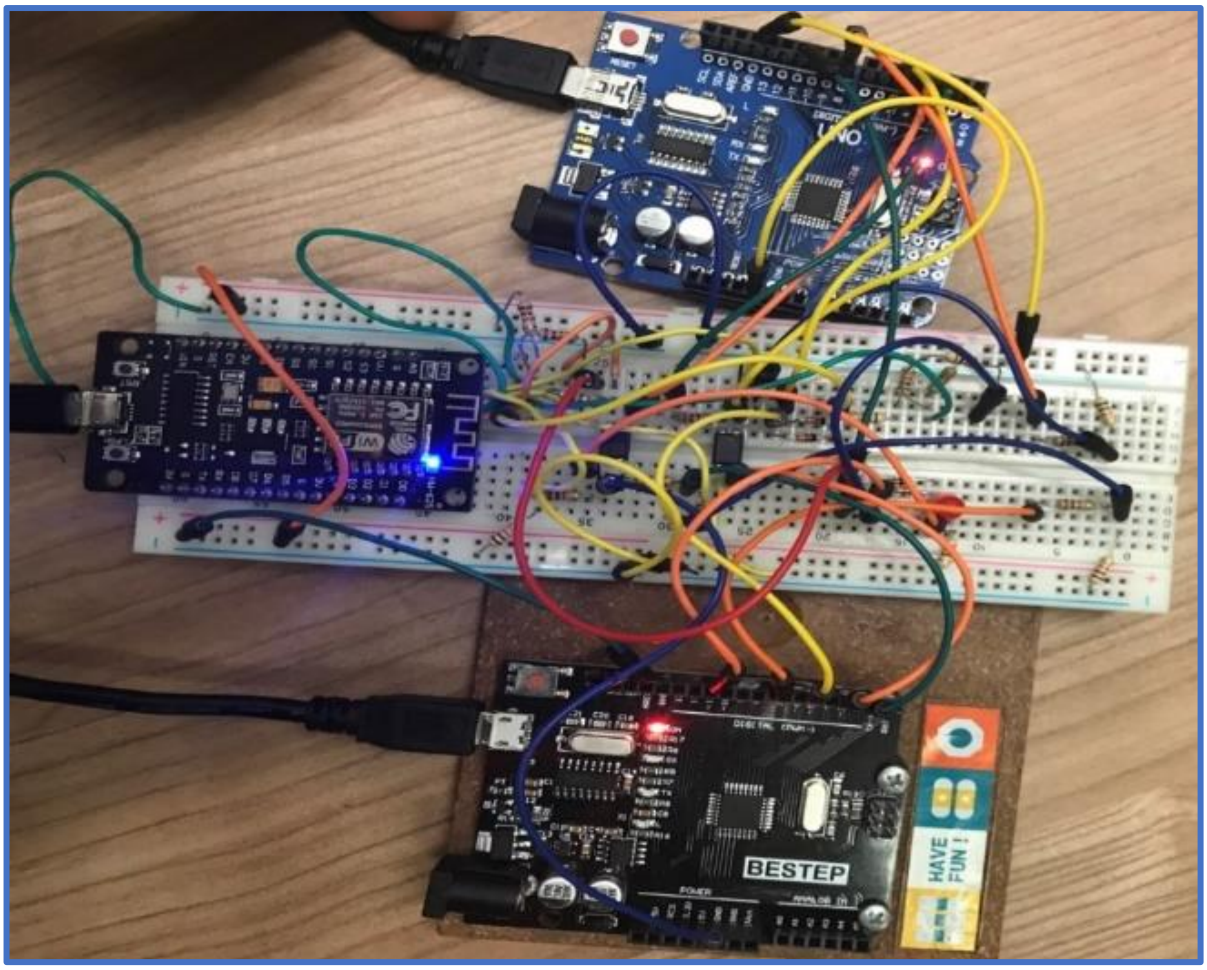
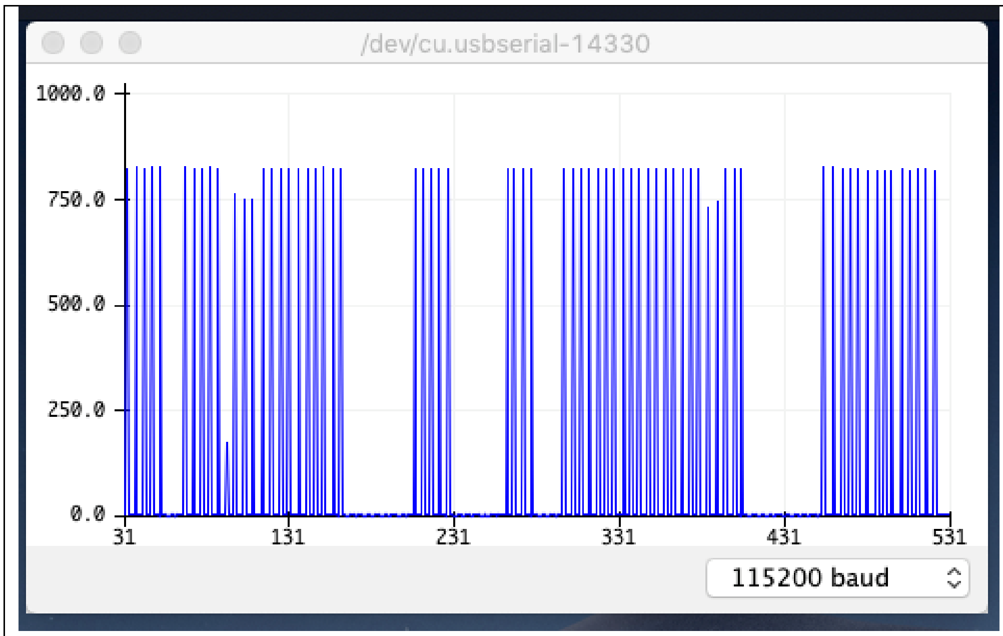
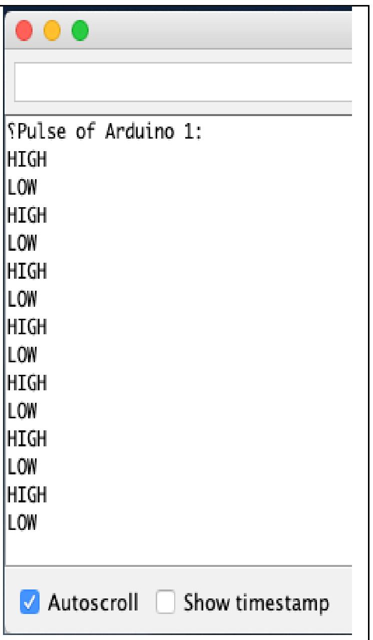


Fig.5 The prototype model of the system

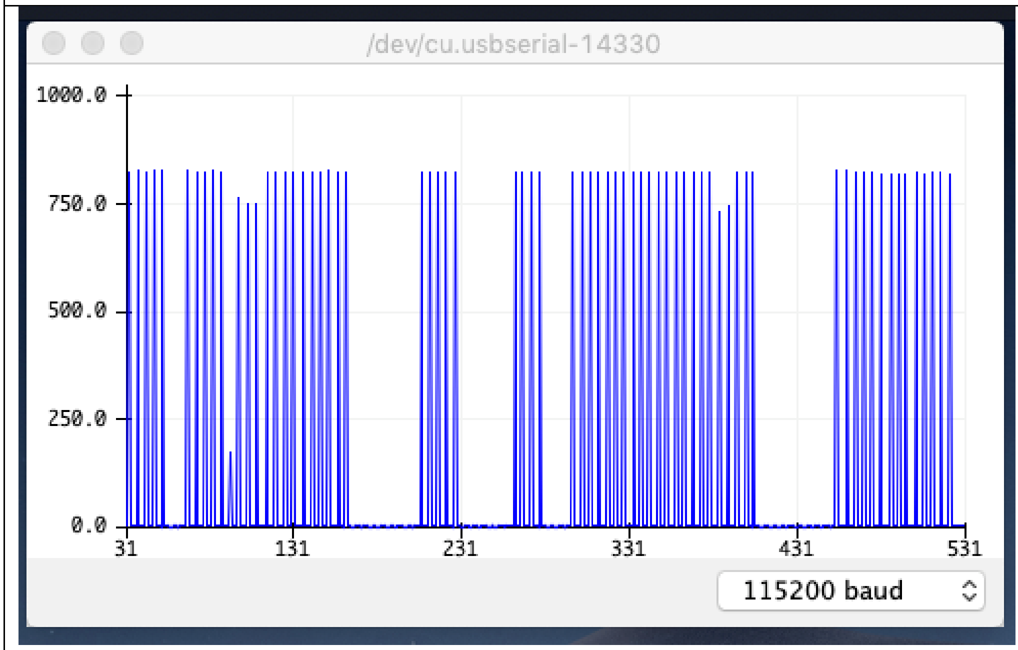
From fig 6. It can be observed that when Arduino 1 is High, Arduino 2 is LOW. This explains the state of the of a masterhood the one in control at that moment. In the case where both are reading HIGH as seen in fig.4 and fig.5, a collision state is attained, which describes the state where both Arduinos are contending to be masters be in control. This is an undesirable situation. Hence a random number is generated continuously which is initiated by the codes as observed in the last system set in methodology to come out of this situation to select a master again(a device that will take over control). Also, it is observed in fig 1. there are signal spikes at some point and none at another location. The signal spikes signify the L.E.D is HIGH (i.e. ON). This shows a master has also been declared, and one of the Arduinos is in control. On the other hand, the ESP2866 module is always also monitoring the voltage of the Arduino through this led to making sure both Arduinos are alive to handle any fault condition. In the code written to run the state machines in the methodology, it will be observed at some point that some lines are indicating Arduino 1 or Arduino 2 is healthy.

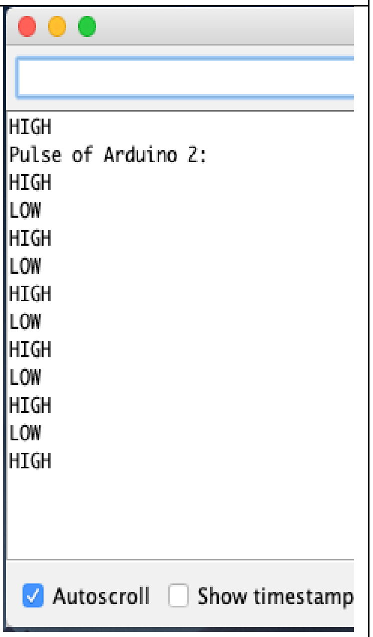
doi:10.20944/preprints201906.0034.v1





```
Pulse of Arduino 1:
HIGH
LOW
HIGH
LOW
HIGH
LOW
HIGH
LOW
HIGH
LOW
HIGH
LOW
HIGH
LOW
```

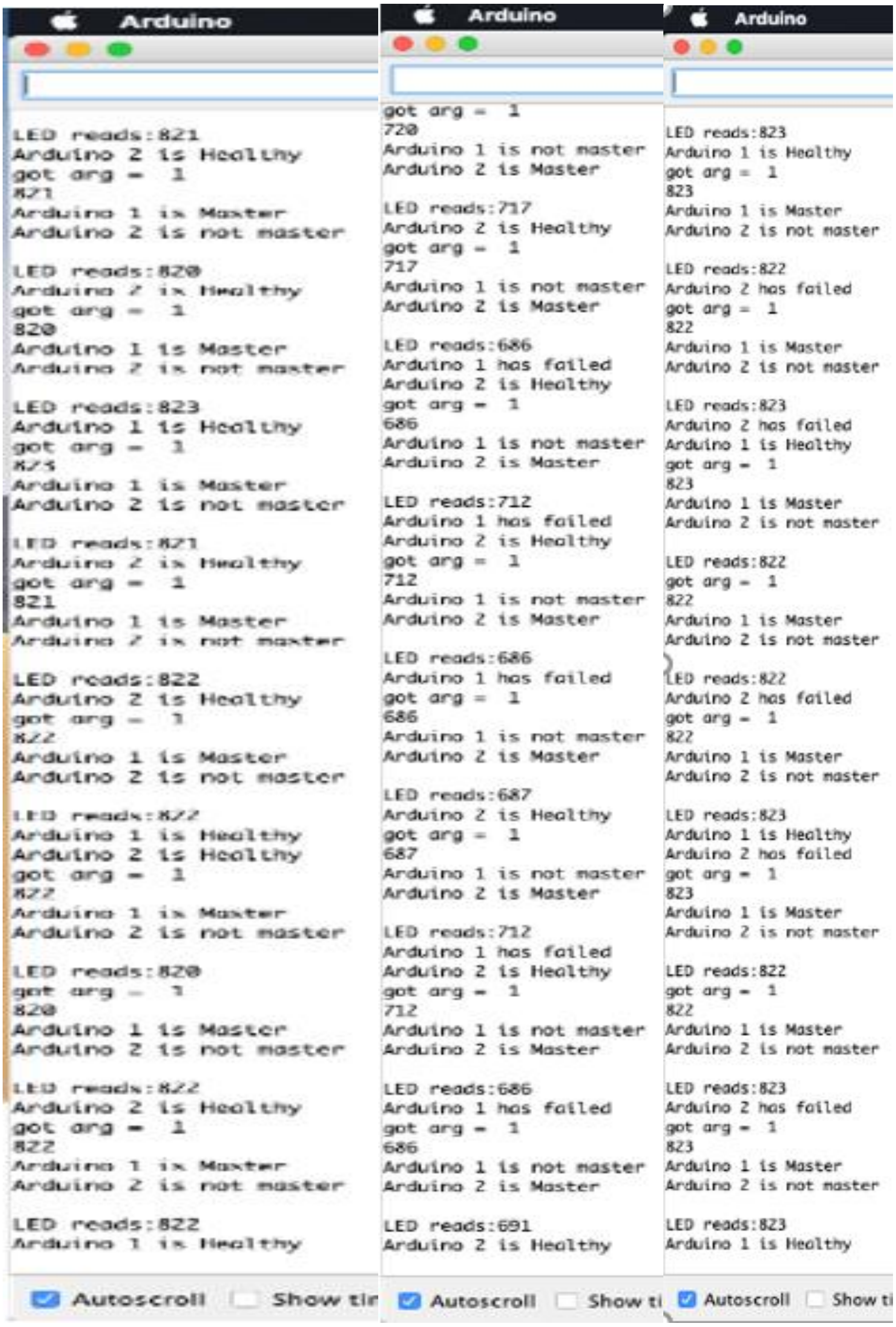




```
HIGH
Pulse of Arduino 2:
HIGH
LOW
HIGH
LOW
HIGH
LOW
HIGH
LOW
HIGH
LOW
HIGH
```

Fig.6 Serial Plotter Reading of the LED State of the system (b) Serial Monitor reading of Arduino pulse (Voltage)

This shows there is no failure and power is flowing correctly through the controller that handles fault conditions. In the advent of one Arduino losing control, the other Arduino will be declared master that is it will act on standby and that Arduino will be indicated as Damaged and unhealthy in the code written to run the state machine as observed in fig 4. Primarily, this study has been able to exemplify a fault tolerant system taking a cue from the response of the Arduino (acting as Signal receivers and fault controllers) at different times the button on the webpage(ESP8266 Wi-Fi module) is pressed. Effectively, when the button is pressed, that is the model of the signal received. When one of the Arduinos indicates a master state, that models a control designation, but when they both show "HIGH" or "LOW" on fig.4 and fig.5. This is the fault condition, is a conflict of interest on which Arduino will be master, hence a master has to emerge using a random number generation code that enforces a stochastic selection process during a time duration to quickly get a master, one of the systems to control.



```
LED reads:821
Arduino 2 is Healthy
got arg = 1
821
Arduino 1 is Master
Arduino 2 is not master

LED reads:820
Arduino 2 is Healthy
got arg = 1
820
Arduino 1 is Master
Arduino 2 is not master

LED reads:823
Arduino 1 is Healthy
got arg = 1
823
Arduino 1 is Master
Arduino 2 is not master

LED reads:821
Arduino 2 is Healthy
got arg = 1
821
Arduino 1 is Master
Arduino 2 is not master

LED reads:822
Arduino 2 is Healthy
got arg = 1
822
Arduino 1 is Master
Arduino 2 is not master

LED reads:822
Arduino 1 is Healthy
Arduino 2 is Healthy
got arg = 1
822
Arduino 1 is Master
Arduino 2 is not master

LED reads:820
got arg = 1
820
Arduino 1 is Master
Arduino 2 is not master

LED reads:822
Arduino 2 is Healthy
got arg = 1
822
Arduino 1 is Master
Arduino 2 is not master

LED reads:822
Arduino 1 is Healthy
```

```
got arg = 1
720
Arduino 1 is not master
Arduino 2 is Master

LED reads:717
Arduino 2 is Healthy
got arg = 1
717
Arduino 1 is not master
Arduino 2 is Master

LED reads:686
Arduino 1 has failed
Arduino 2 is Healthy
got arg = 1
686
Arduino 1 is not master
Arduino 2 is Master

LED reads:712
Arduino 1 has failed
Arduino 2 is Healthy
got arg = 1
712
Arduino 1 is not master
Arduino 2 is Master

LED reads:686
Arduino 1 has failed
got arg = 1
686
Arduino 1 is not master
Arduino 2 is Master

LED reads:687
Arduino 2 is Healthy
got arg = 1
687
Arduino 1 is not master
Arduino 2 is Master

LED reads:712
Arduino 1 has failed
Arduino 2 is Healthy
got arg = 1
712
Arduino 1 is not master
Arduino 2 is Master

LED reads:686
Arduino 1 has failed
got arg = 1
686
Arduino 1 is not master
Arduino 2 is Master

LED reads:691
Arduino 2 is Healthy
```

```
LED reads:823
Arduino 1 is Healthy
got arg = 1
823
Arduino 1 is Master
Arduino 2 is not master

LED reads:822
Arduino 2 has failed
got arg = 1
822
Arduino 1 is Master
Arduino 2 is not master

LED reads:823
Arduino 2 has failed
Arduino 1 is Healthy
got arg = 1
823
Arduino 1 is Master
Arduino 2 is not master

LED reads:822
got arg = 1
822
Arduino 1 is Master
Arduino 2 is not master

LED reads:822
Arduino 2 has failed
got arg = 1
822
Arduino 1 is Master
Arduino 2 is not master

LED reads:823
Arduino 1 is Healthy
Arduino 2 has failed
got arg = 1
823
Arduino 1 is Master
Arduino 2 is not master

LED reads:822
got arg = 1
822
Arduino 1 is Master
Arduino 2 is not master

LED reads:823
Arduino 2 has failed
got arg = 1
823
Arduino 1 is Master
Arduino 2 is not master

LED reads:823
Arduino 1 is Healthy
```

Fig. 7 Serial Monitor Reading of the Arduino State

5. Conclusions

A fault tolerant system was built, using an Arduino micro-controller to implement a random decision-making system in the advent of failure in any order. Even though the used set-up is relatively simple, it was shown to be able to iteratively decide without any human intervention in any chosen circumstance of reference in which in this case it is the push of a button. The result obtained shows that the aim that the system can be designed to be a fault tolerant system with a single point of failure provides enough facts for an accurate system application. However, one drawback of this current system, and similar methods of this type, in general, is that great care and patience must be taken during testing to test all components that make up the system to ascertain they are all electrically conducting. This necessitates the use of a multimeter to prevent confusion as to where there is a fault when final testing of the final test design is to be done. Further improvement of the system can be made by viewing the signals of the respective component of the order on the serial monitor of the Arduino integrated environment to see which part of the system. It is dormant and might probably inhibit the final design from working accordingly and also introducing fuzzy logic to the model may boost its functionality and application.

REFERENCES

[1]. wiki, “Wikipedia,” Wikipedia incorporation, 10 November 2018. [Online]. Available: https://en.wikipedia.org/wiki/Grenfell_Tower_fire. [Accessed 20 October 2018].

[2]. L. Tae-Soo, H. Joo-Hyun, and C. Myeong-Chan, "Biomedical Digital Assistant for Ubiquitous Healthcare," in Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE, 2007, pp. 1790-1793.

[3]. N. H. Mahmood, N. Uyop, N. Zulkarnain, F. K. C. Harun, M. F. Kamarudin, and A. Linoby, "LED indicator for heart rate monitoring system in sport application," in Signal Processing and its Applications, 2011 IEEE 7th International Colloquium on, 2011, pp. 64-66.

[4]. V.A. Baloguna, B.I. Oladapo, A.O.M. Adeoye, J.F. Kayode, S.O. Afolabi, Hysteresis analysis of Thornton (IP6, IP12E and TH5V) magnetic materials through the use of arduino microcontroller, J. Mater. Res. Technol. (2018), http://dx.doi. org/10.1016/j.jmrt.2017.05.018.

[5]. N. S. A. Zulkifli, F. K. C. Harun, and N. S. Azahar, "XBee wireless sensor networks for Heart Rate Monitoring in sport training," in Biomedical Engineering (ICoBE), 2012 International Conference on, 2012, pp. 441-444.

[6]. A.O.M. Adeoye, J.F. Kayode, B.I. Oladapo, S.O. Afolabi, Experimental analysis and optimization of synthesized magnetic nanoparticles coated with PMAMPC-MNPs for bioengineering application, St. Petersburg Polytech. Univ. J. 3 (4) (2017) 333–338.

[7]. Dave Locke , MQ telemetry transport (mqtt) v3. 1 protocol specification. IBM developer Works Technical Library] , 2010 [online] Available: http://www.ibm.com/developerworks/webservices/library/ws-mqtt/

[8]. M. Panda, P.M. KhilarDistributed self fault diagnosis algorithm for large scale wireless sensor networks using modified three sigma edit test Ad Hoc Netw., 25 (2015), pp. 170-184

[9]. L. Tae-Soo, H. Joo-Hyun, C. Myeong-Chan Biomedical Digital Assistant for Ubiquitous Healthcare in Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE (2007), pp. 1790-1793

[10]. Liou J.J., Tzeng G.-H., Chang H.-C. Airline safety measurement using a hybrid model J. Air Transp. Manage., 13 (2007), pp. 243-249.

[11]. M. Hennessy, R. Milner Algebraic laws for nondeterminism and concurrency J. ACM, 32 (1) (1985), pp. 137-161.

[12]. Bankole. I. Oladapo, S. Abolfazl Zahedi, Surya C. Chaluvadi, Satya S. Bollapalli, Muhammad Ismail. Model design of a superconducting quantum interference device of magnetic field sensors for magnetocardiography. Biomedical Signal Processing and Control 46 (2018) 116–120

[13]. Abdel-Basset M., Manogaran G., Gamal A., Smarandache F.A hybrid approach of neutrosophic sets and DEMATEL method for developing supplier selection criteria Des. Autom. Embedded Syst. (2018), pp. 1-22.

[14]. A.O.M. Adeoye, B.I. Oladapo, A.A. Adekunle, A.J. Olademeji, J.F. Kayode, Design, simulation and implementation of a PID vector control for EHVPMSM for an automobile with hybrid technology, J. Mater. Res. Technol. (2018), http://dx. doi.org/10.1016/j.jmrt.2017.07.005

[15]. N.S.A. Zulkifli, F.K.C. Harun, N.S. Azahar XBee wireless sensor networks for Heart Rate Monitoring in sports training Biomedical Engineering (ICoBE), 2012 International Conference on (2012), pp. 441-444.

[16]. W. v. Drongelen, Signal Processing for Neuroscientists (Second Edition), Chicago, IL, USA: Elsevier, 2018.

[17]. B.I. Oladapo, A.O.M. Adeoye, S.O. Oyegoke, S.O. Afolabi, Analysis for distortion of thin-wall milling on machine component and its effect on global warming, Procedia Manuf. 7 (2017) 529–536.

[18]. c. Wires, "Google," carstool.ru. Available: http://carstool.ru/nabor-tectovix-provodov-dlya-mul-timetra-banan-krokodil. [Accessed 5th May 2019].

[19]. A. Industries, "Google," Adafruit Industries. Available: https://www.adafruit.com /product/2781. [Accessed 6th May 2019].

[20]. r. Capacitors, "capacitors,". Available: https://www.radioshack.com/products/radioshack-0-1uf-50v-hi-q-ceramic-disc-capacitor-pk-2. [Accessed 5th May 2019].

[21]. Arduino,"ARDUINO," 2018. Available: https://www.arduino.cc/en/Guide/Introduction. [Accessed 5 May 2019].