

INFT3970 Major Project Final Report

Distributed Monitoring System using ESP-12E

Team Encore

Thursday 10:00AM - 10:30AM

Jay Rovacsek c3146220@uon.edu.au

Dean Morton c3252227@uon.edu.au

Josh Brown c3283797@uon.edu.au

Jacob Litherland c3263482@uon.edu.au

Lee Marron c3263482@uon.edu.au

Edward Lonsdale c3252144@uon.edu.au

October 22, 2018

Contents

1	Executive Summary	2
2	Introduction	3
2.1	Project Objectives	3
2.2	Business Summary	3
3	Proposed Solution	4
3.1	Outline	4
3.2	Technical Requirements	4
3.3	Limitations or Constraints	4
4	Technical Documentation	5
4.1	Database Documentation	5
4.1.1	Relational Model	5
4.1.2	Data Requirements	5
4.1.3	Data Dictionary	5
4.1.4	Database Schema	5
4.2	ESP12-E Documentation	6
4.2.1	Temperature	6
4.2.2	Humidity	6
4.2.3	Motion	6
4.2.4	Issues in Development	6
4.3	Web Application Documentation	7
4.3.1	API Documentation	7
4.3.2	Issues in Development	7
4.4	General Documentation	8
4.4.1	Sequence Diagrams	8
4.4.2	Network Diagram	8
4.5	Install Procedures	9
5	User Documentation	10
5.1	Installation	10
5.2	Accessing Dashboards	10
5.3	Account Administration	10
A	Appendix	12
A.1	Golang ESP12-E Implementation	12

1 Executive Summary

2 Introduction

Sensemaking is the process by which businesses give meaning to their collective experiences. It is often formally defined as "the ongoing retrospective development of plausible images that rationalize what people are doing" [1]. The concept was intended to encourage a shift to the meaning of the decisions. As human beings traditional senses are limited in the face of infinite complexity. With the advent of modern science in particular high scale integration humans were able to create artificial sensors that increase their capabilities for data collection. The problem then became how do use this data, and the huge broad or deep datasets created by these data new streams. In the current year we share a cybernetic relationship with these technologies. Sensor technology is being used in transport systems, healthcare, education and even in handheld phones.

Technology is fast becoming an extra sense and in that vein it is becoming seminal to understand the technology, the data its produced and the potential for aggregation into decision making knowledge.

2.1 Project Objectives

2.2 Business Summary

As discussed in the introduction modern businesses have the opportunity to potentially access massive datasets. This data can be overwhelming and difficult to visualise and use in the process of sensemaking. The business purpose of this project is to take a environment install sensors record the data, use modern technology to aggregate and visualise data for sensemaking.

3 Proposed Solution

3.1 Outline

3.2 Technical Requirements

3.3 Limitations or Constraints

4 Technical Documentation

4.1 Database Documentation

4.1.1 Relational Model

4.1.2 Data Requirements

4.1.3 Data Dictionary

4.1.4 Database Schema

4.2 ESP12-E Documentation

4.2.1 Temperature

4.2.2 Humidity

4.2.3 Motion

4.2.4 Issues in Development

One of the first challenges of this project was to ensure we had a viable path to producing a result we deemed desirable, a rused prototype was created on the ESP12-E using arduino code. The code used methods causing blocks on continued execution and for one or two items being reported this was okay, however to ensure the platform had room to breathe we investigated two routes: non-blocking code using timers in arduino code, and golang utilising the Gobot framework[2].

The Gobot route seemed very promising, prewritten libraries for items such as the XC-4444[3] existed, however a mixture of team members unfamiliar with the Go language[4] and a number of resources pointing to the DHT-11[5] being severely undersupported[6]. in the Golang space lead us to investigate the alternative option: timers in arduino code.

A copy of the Golang Arduino Implementation can be found [here](#).

4.3 Web Application Documentation

4.3.1 API Documentation

4.3.2 Issues in Development

4.4 General Documentation

4.4.1 Sequence Diagrams

4.4.2 Network Diagram

4.5 Install Procedures

5 User Documentation

5.1 Installation

5.2 Accessing Dashboards

5.3 Account Administration

References

- [1] K. E. Weick, K. M. Sutcliffe, and D. Obstfeld. (2005). Organizing and the process of sensemaking.
- [2] Various. (2018). The gobot framework, [Online]. Available: <https://gobot.io/>.
- [3] —, (2018). Jaycar xc-4444 datasheet, [Online]. Available: https://www.jaycar.com.au/medias/sys_master/images/9105858396190/XC4444-dataSheetMain.pdf.
- [4] —, (2018). The golang language, [Online]. Available: <https://golang.org/>.
- [5] —, (). Jaycar resources; dht11 datasheet, [Online]. Available: https://www.jaycar.com.au/medias/sys_master/images/9091897786398/XC4520-dataSheetMain.zip.
- [6] —, (2018). Issues implementing dht-11 and dht-22, [Online]. Available: <https://github.com/hybridgroup/gobot/issues/361>.

A Appendix

A.1 Golang ESP12-E Implementation

```
1 package main
2
3 import (
4     "C"
5     "dht"
6     "fmt"
7     "net/http"
8     "time"
9
10    "gobot.io/x/gobot"
11    "gobot.io/x/gobot/drivers/gpio"
12    "gobot.io/x/gobot/platforms/firmata"
13 )
14
15 const (
16     SensorDHT11 = iota
17     SensorDHT22
18 )
19
20 var firmataAdaptor = firmata.NewTCPAdaptor("192.168.1.95:3030")
21
22 func main() {
23
24     work := func() {
25         gobot.Every(1*time.Second, func() {
26             tryFlashLED()
27         })
28     }
29
30     led := gpio.NewLedDriver(firmataAdaptor, "2")
31
32     robot := gobot.NewRobot("bot",
33         []gobot.Connection{firmataAdaptor},
34         []gobot.Device{led},
35         work,
36     )
37
38     robot.Start()
39 }
40
41 func tryFlashLED() {
42     var available = getAvailability('http://inft3970.azurewebsites.net
43         :80/api/Availability')
44
45     fmt.Println(available)
46
47     if available {
48         flashLED('100ms', 10)
49         return
50     }
51 }
```

```

49     }
50
51     flashLED('500', 2)
52     return
53 }
54
55 func flashLED(milliseconds string, iterations int) {
56     led := gpio.NewLedDriver(firmataAdaptor, "2")
57
58     duration, _ := time.ParseDuration(milliseconds)
59
60     for i := 1; i <= iterations; i++ {
61         led.Toggle()
62         time.Sleep(duration)
63         led.Toggle()
64     }
65 }
66
67 func getAvailability(server string) bool {
68
69     response, err := http.Get(server)
70
71     var humidity, temperature, _ = getSensorData(SensorDHT11, 2)
72     if err != nil {
73         printError(err)
74     }
75
76     fmt.Println(fmt.Sprintf("Temperature: %f, Humidity: %f"), temperature
77         , humidity)
78
79     if err != nil {
80         printError(err)
81     }
82
83     if response.StatusCode == 200 {
84         return true
85     }
86     return false
87 }
88 func getSensorData(stype, pin int) (humidity, temperature float32, err
89     error) {
90     if stype != SensorDHT11 && stype != SensorDHT22 {
91         err = fmt.Errorf("sensor type must be either %d or %d",
92             SensorDHT11, SensorDHT22)
93         return
94     }
95
96     var data [5]byte
97     data, err = dht.ReadSensor(pin)
98     if err != nil {
99         return
100     }

```

```

99
100     if stype == SensorDHT11 {
101         humidity = float32(data[0])
102         temperature = float32(data[2])
103     } else {
104         humidity = float32(int(data[0])*256+int(data[1])) / 10.0
105         temperature = float32(int(data[2]&0x7F)*256+int(data[3])) / 10.0
106         if data[2]&0x80 > 0 {
107             temperature *= -1.0
108         }
109     }
110     return
111 }
112
113 func printError(err error) {
114     fmt.Println(fmt.Sprintf("An error occured: %v", err.Error()))
115 }

```
