

## WORCESTER POLYTECHNIC INSTITUTE

ECE 2305 – FINAL PROJECT PROPOSAL  
SPAS (SEARCH AND PICK A SEAT), GO ONLINE AND PICK  
YOUR TABLE DESK (GOAT) GOAD

*Anastasia Karapanagou (640938922) (25%)*

*Fivos Kavassalis (124919167) (25%)*

*Ioannis Alexiou (631078066) (25%)*

*Josue Contreras (851386562) (25%)*

Supervised by

Dr. Alexander Wyglinski

 [https://github.com/FivosJKavassalis/ECE2305\\_Team14](https://github.com/FivosJKavassalis/ECE2305_Team14)

 <https://www.youtube.com/watch?v=R3chP828VOM&t=15s>

May 1, 2018

## **Table of Contents**

Abstract.....	2
Introduction.....	2
Description of Proposed Solution and Brief Overview of Implementation.....	2
Experimental Results.....	7
Summary of Challenges Encountered and Lessons Learned.....	13
Discussion.....	13
Conclusion.....	14
Appendix.....	14

## **Abstract**

An IoT system was designed and implemented that comprises a wireless network capable of connecting multiple low complexity devices (sensors) to the Internet. The objective was to obtain information about which Gordon Library cubicles are empty, and therefore available for use, at any particular time. Each cubicle possessed a simple sensor connected to an embedded processor. The desired task was to detect whether the cubicle was occupied or not. This information would be transmitted via the Internet to the members of the WPI community seeking an empty cubicle space to work. Our system addresses the need of the WPI community for efficient use of time and resources, especially in periods of high demand for quiet space in the library.

## **Introduction**

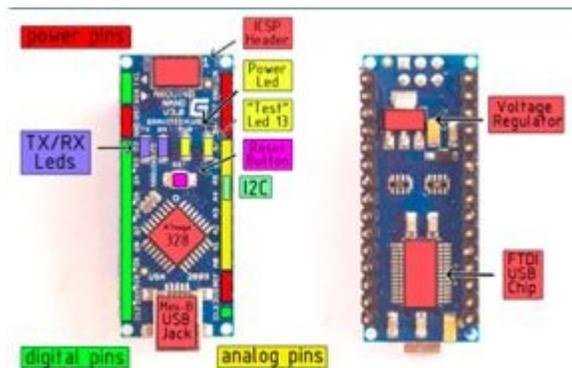
The Gordon Library, which offers space to students that wish to find a quiet place to study, is one of the most popular places on the WPI campus. Unfortunately, at peak times, students have a hard time finding a seat in the cubicle desk spaces located in the library. As the population of both undergraduate and graduate students on campus increases, demand for space on campus increases as well. Usually, most students seek places around campus to study and focus on work but looking for space can be a waste of time and energy. This is a great problem for WPI students' tight schedules. Efficient use of time in a busy and demanding academic environment is fundamental for the timely completion of work requirements, for students and faculty alike.

The team, therefore, decided to design a system by which any member of the WPI community may know at any given time whether the particular place/environment they are looking for, is occupied or empty, and, in this way, make an informed decision about its availability and use. Our project was designed to identify empty cubicle desk spaces in the Gordon Library and communicate it to the WPI community via the Internet. The team designed and implemented a wireless network capable of connecting multiple sensors to the Internet in order to obtain information about which desk cubicles are empty, therefore available to use, at any particular time. We feel we have contributed in this way towards addressing an important need for the WPI community.

## **Description of Proposed Solution and Brief Overview of Implementation**

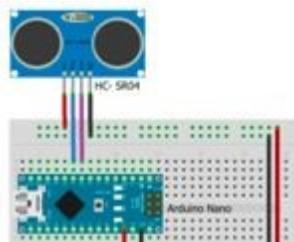
### **STEP 1: Set-up of the Physical Layer**

For data processing, we decided to use the Arduino Nano V3.0 and Arduino Uno microcontrollers due to their small dimensions and cost-effectiveness. Specifically, the Arduino Nano interfaced with the ultrasonic sensor HC-SR04 we decided to include. This microcontrollers' array has a variety of ports to choose from so we could supplement our design with additional modules (shown in Figure 1).



**Figure 1:** Arduino Nano V3.0 Microcontroller.

The first step involved is the interface of the microcontroller with the HC-SR04 ultrasonic sensor(s) (shown in Figure 2). The ultrasonic sensor operates at a frequency of 40KHz and has a range from 2 cm to 4 m which is suitable for our system. The status of the cubicle was reported every T seconds. We designed it according to the TTL (Transistor-Transistor Logic) communication protocol so that by a 10us pulse (high state) to the trigger pin, the module sent out an 8-cycle sonic burst. Then, the echo pin received this burst and outputted the time in microseconds. We have employed the following equation (shown in Figure 3) to find the distance between an object and the ultrasonic sensor, which helps us understand whether the cubicle is occupied.



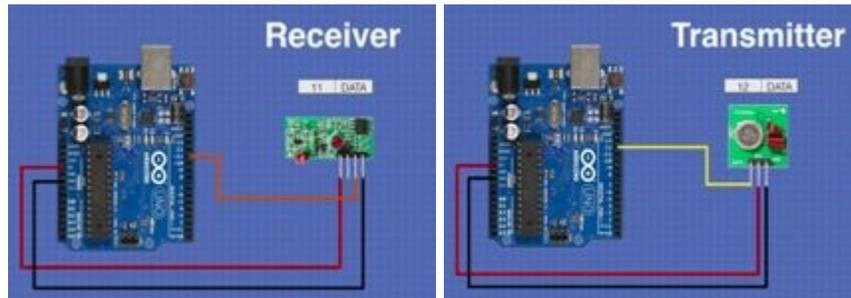
**Figure 2:** HC-SR04 Ultrasonic Sensor

$$\text{distance} = (\text{duration} * 0.034) / 2$$

**Figure 3:** HC-SR04 Ultrasonic Sensor Distance Equation

Our sensor communicates to the aggregator node via the 433MHz RF front-end RX and TX modules. The transmitter possesses three pins: GND, Vcc and DATA. We used the DATA pin that is connected to a Digital pin on the arduino to transmit the signal, in this case the sensor data, to our receiver. On the other hand, the receiver possesses four pins: GND, Vcc and two DATA pins. We used only one DATA pin (shown in Figure 4). An antenna allowed our system to propagate the signal at greater distances, therefore solving any attenuation/path loss that the signal could encounter. However, we were also careful not to overamplify the signal as this could cause interference with other signals. We also

realized that we had to optimize the various antenna configurations and distances ourselves. We used a coiled-up antenna and 17.3 cm of straight antenna to the receiver and the transmitter. The ASK (Amplitude Shift Keying) employed which controls the carrier wave was encoded and decoded but also limited the amount of data communicated. Another issue we worked on was the reduction of noise. We addressed this by using the Radiohead library.



**Figure 4:** Data Pin on Receiver (left) and Transmitter (right).

The physical layer was completed by the addition of an LCD and LEDs for debugging purposes. For the former, the Crystal library was used to show the status of the modules and whether data is being received. The LEDs served as indicators that the modules are working correctly.

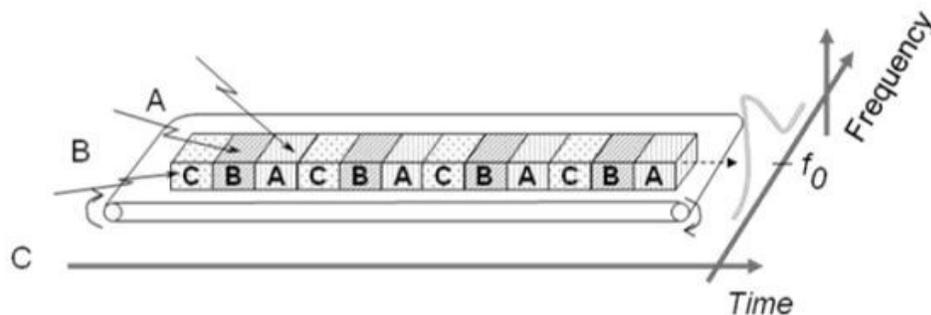
## STEP 2: Medium Access Control Layer

We proceeded by working on the Medium Access Control Layer to be implemented in Arduino. To identify the sensor sending data and its location, we had to frame a datagram and add overhead recognizing the sensor number and the embedded processor name (Figure 5). The sender, in this case the sensor node, added only its sensor number node as overhead. This way when the aggregator node received that data, it knew how to filter out the data into the right channels. Each message sent out was a nine-character long string. The first two letters of the string were the name, for example “N1” for node one. Then, the next seven characters were either one for an occupied seat, zero for a free seat, or a comma in order for the aggregator to know where to separate the data.

- Process
  - Sender
    - ➊ Receive datagram from network layer
    - ➋ Encapsulates datagram into frame
    - ➌ Adds error checking bits, RDT, flow control, etc
  - Receiver
    - ➊ Receives frames from physical layer
    - ➋ Looks for errors, RDT, flow control, etc
    - ➌ Extracts datagram, passes to upper layer (network layer) at receiving side

**Figure 5:** Datagram Construction and Deconstruction

This information was then sent to the aggregator node where the datagram was filtered through the right channels to be executed. By correlating data with identifiable sensors and embedded processors, we were able to test the reliability of our set-up and choose our data display. In addition, we had to consider how we would address handling multiple sensor nodes communicating with one aggregator node. We reasoned we can solve this either by the Time Division Multiple Access (TDMA) or by the Code Division Multiple Access (CDMA) approach. We chose to use the TDMA. This channel access method divides the signal into different time slots, users transmit rapidly one after the other while using a single frequency (433 MHz) and in this way, use only part of the available capacity. In our case, only one sensor node was able to transmit during each time slot. The aggregator served as a master that transmitted the sensor node name that it wanted to communicate with. Then, the corresponding sensor node sent out the frame to the aggregator and once it finished processing the data, the aggregator node called on the next sensor node it wanted to communicate with. By using multiple time slots, the system communicated an increased amount of data. Its implementation required that our sensor nodes and aggregator node have a pair of transmitter and receiver modules for each node. Additionally, the system was able to check for corrupted files and data not received with a parity checking process.

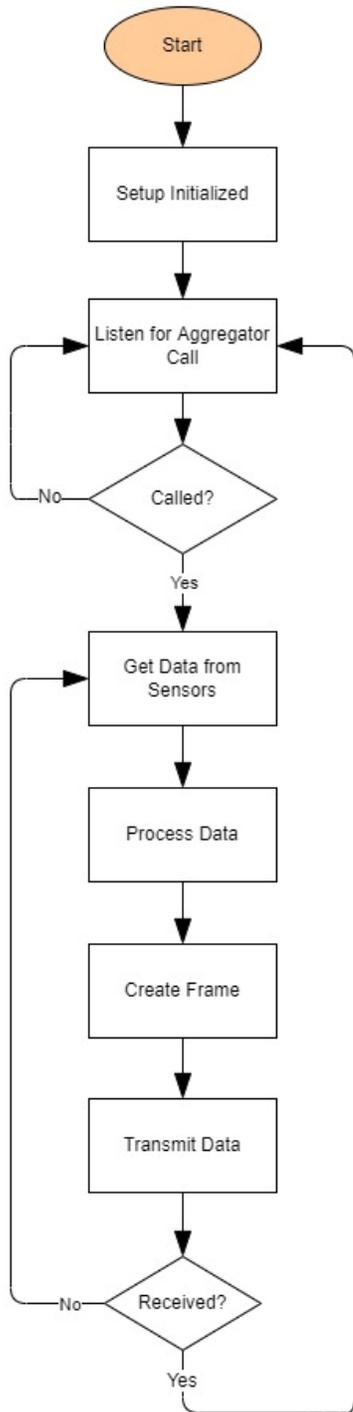


**Figure 6:** Simplified Diagram of TDMA Showing 3 Users Transmitting in Succession, first A, then B, finally C and so on (from [https://www.tutorialspoint.com/cdma/tdma\\_technology.htm](https://www.tutorialspoint.com/cdma/tdma_technology.htm))

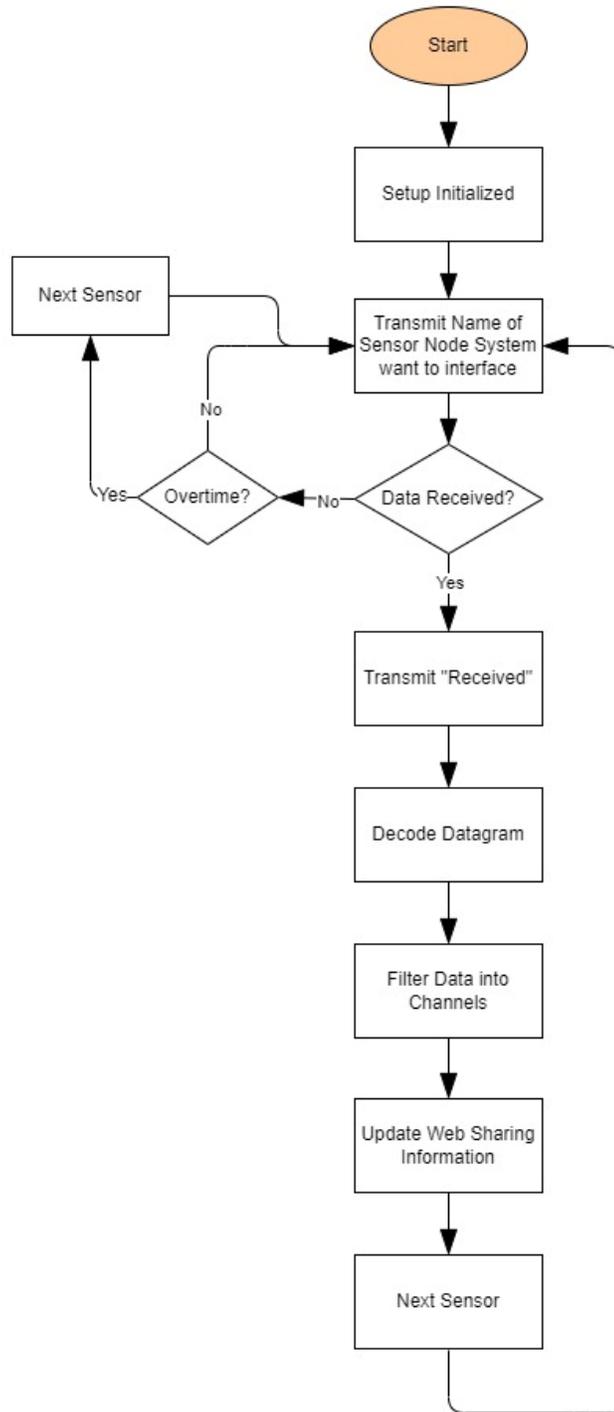
As with every system, there are advantages (as outlined above) and disadvantages with TDMA. Some of the caveats involve the fact that if time slots are short, then complicated signal processing should be employed, and complex equalization may be needed.

If we had opted for the CDMA method, we would have added complexity to the code implementation of the aggregator node as the aggregator would have to focus only on one sensor node transmitting. In this way, ideally, we could implement transmitters on the sensor nodes and one receiver on the aggregator node, thus enabling us to have an error/corruption checking scheme, namely the CRC scheme. This option comes with the caveat that data cannot be re-transmitted because of the one transmitter-one receiver implementation mode.

## Sensor Node Flowchart



## Aggregator Node Flowchart



**Figure 7:** Sensor Node (left) and Aggregator Node (right) Flowcharts.

### STEP 3: Connectivity to the Internet and Web Sharing Information

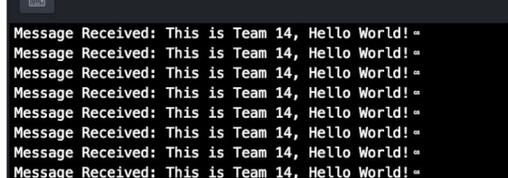
For the connectivity step, we used the Blynk Library with a unique authentication token to interface between the Arduino Uno and our smartphone application. Our effort was directed towards taking data from the Application layer of the aggregator and sending it to a user across the Internet. The Blynk Application would allow us to display our data, like the occupancy of cubicles in various areas of the library, in a user-friendly graphical user interface (GUI),

### Experimental Results

Throughout implementation of this IoT device we made sure to test everything we added carefully, but due to WPI's fast-paced 7-week term and other classes we were not able to test the final product in its entirety.

We started this project by testing out the one-way communication between our Tx and Rx modules. This can be seen by the successful message in the figure below. Something interesting that our team did not notice at this stage of our project was the “garbage” that can be seen attached after the message received. This gave us problems during the implementation and will be discussed later in this paper.

```
65
66 void loop()
67 {
68   // Set buffer to size of expected message
69   uint8_t buf[29];
70   uint8_t buflen = sizeof(buf);
71   // Check if received packet is correct size
72   if (rf_driver.recv(buf, &buflen))
73   {
74     // Message received with valid checksum
75     Serial.print("Message Received: ");
76     Serial.println((char*)buf);
77   }
78 }
79
```



```
Message Received: This is Team 14, Hello World! =
Message Received: This is Team 14, Hello World! =
Message Received: This is Team 14, Hello World! =
Message Received: This is Team 14, Hello World! =
Message Received: This is Team 14, Hello World! =
Message Received: This is Team 14, Hello World! =
Message Received: This is Team 14, Hello World! =
Message Received: This is Team 14, Hello World! =
```

Figure 8: The first message received to test all the Tx and Rx modules

During this stage we also tested the communication range of the modules. Our results showed that 8 feet was the maximum range before the packages started dropping at a fast rate. We also must take into account that this was a controlled, peaceful environment, which is the complete opposite from where this device would be located, which is the

library. We noticed that as a human body got closer to the modules, they would drop packages even more and would lose connection rapidly. It was interesting that when the receiver was pointed towards (this being the green box part as the front of the transmitter) the receiver did not receive the signal. However, when pointed the other way or sideways, it would receive the signal. This experiment was done in Line of Sight (LOS), meaning without any obstruction, because of the attenuation/path loss as the signal traveled from the Tx to the Rx module.



**Figure 9:** Range without antenna test

Finally, during this stage, we tested out the communication between multiple transmitters and one receiver even though we knew there was going to be collision. We got the following results: When transmitting two sensors at a time without any protocol, the serial port immediately stopped printing values that were being transmitted. This meant that because two signals were transmitting at the same time and in the same frequency, collision occurred at the receiver module. After turning off one of the sensor nodes, it took an average of 2 seconds to start transmitting again, considering that there was no error checking or protocol.

On the next stage of the implementation process, we added the antennas. We decided to use a 17.3 cm antenna on both the receiver and transmitter modules. We came to this length by taking a fraction of the 433MHz wavelength (Figure 10).

$$Speed = \frac{\text{Distance traveled}}{\text{Time taken}} = \frac{\text{Wavelength}}{\text{Period}} = \left(\frac{1}{\text{Period}}\right) \times \text{Wavelength}$$

$$Speed = \text{Frequency} \times \text{Wavelength}$$

**THE SPEED OF A WAVE**

$$\text{Speed (m/sec)} \rightarrow v = f \lambda$$

Frequency (hertz)  $f$   
Wavelength (meters)  $\lambda$

**Figure 10:** Wavelength equation

After adding an antenna, transmission distance increased to about 19.812 meters (Figure 11). As shown in figure 12, we were able to successfully transmit and receive messages at that distance. However, this was before the transmission started slowing down and losing packets.



**Figure 11:** Transmission physical distance with Antenna LOFS

```
C:\main.cpp -- ECE2305 - IoT Code/src  Pins.h  C:\main.cpp -- Tx_module_code/src
21 // Create Amplitude Shift Keying Object
22 RH_ASK_rf_driver;
23
24 void setup()
25 {
26 // Initialize ASK Object
27 rf_driver.init();
28 // Setup Serial Monitor
29 Serial.begin(9600);
30 }
31
32 void loop()
33 {
34 // Set buffer to size of expected message
35 // Antenna Test successful,
36 uint8_t buf[42];
37 uint8_t buflen = sizeof(buf);
38 // Check if received packet is correct size
39 if (rf_driver.rcv(buf, &buflen))
40 {
41 // Message received with valid checksum
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
262
```

Regarding the Multiple Access Layer and Link Layer experimental results, we ended up not doing error detection since it was included in the Radiohead library. When running the system, we had a 66% success rate meaning that only 32% of errors passed. This is not the best, but for this system it works out perfectly fine. Most of the packets dropped where when one sensor node would take time to transmit its sensor readings and would go over its allocated time. There was sometimes uncertainty towards what sensor readings the aggregator node would receive. There were cases where the aggregator node would receive the same sensor data twice in a row. Even though this happened, we found out that because the Web Layer did not update immediately, the false readings did not show up that many times. This is important because that way the users can receive the right readings most of the time. Additional features could be added to the code in order to encapsulate these edge cases.

Furthermore, our datagram changed, and instead of the ultrasonic sensor number, we used the microprocessor name. We did this because during our experimentation process, we realized that as the package was smaller the signals dropped would be less. We then realized that we could further implement a way of using hex values in order to transmit smaller packages for better communication.



**Figure 13:** Original datagram

Next, we worked on the Web Sharing Layer. We first tried to use Google Docs to print the messages, but we were unable to since it printed readings from only two sensor nodes instead of 4, which is what we wanted. Therefore, we ended up using the Blynk app, which had the advantage of printing messages on the phone. Blynk uses an authentication token and is very easy to use and setup. To install and set up Blynk, the following steps were taken:

- First, install the libraries on Arduino
- Second, run on terminal the following commands (mac version)
  - Go to your arduino libraries through terminal
  - `cd Arduino/`
  - `Cd libraries/`
  - `Cd Blynk/`
  - `Cd scripts`
  - `./blynk-ser.sh`
  - `brew install socat`
  - Then select the right port and it should start working
- Third, open app and press the play button

During our implementation, we encountered a transmission issue. When the sensor node first receives the message it authenticates it perfectly, but in the second iteration it does not. There seems to be something added which prevents correct authentication (Figures 14 and 15).

```
cpf
h
Sensor Node 1 Listening...;)
Authenticating Number...
1
Transmitting...
N1,0,0,0,
Transmitting...
N1,0,0,0,
Transmitting...
N1,0,0,0,
Transmitting...
N1,0,0,0,
Transmitting...
N1,0,0,0,
```

**Figure 14:** Message received when transmitting correctly.

```
Authenticating Number...
Not me...
1rx
Sensor Node 1 Listening...;)
```

**Figure 15:** Second Iteration problem.

The maximum sensor nodes per aggregator node were 9. This restriction was because of how the code uses the `atoi()` function to convert an integer to a char. Further implementation could help with expanding the ratio of sensor nodes per aggregator nodes, but for this system's purposes, 9 is more than sufficient.

On the last stage of the implementation process, we tested out various API's. We tried to implement the web layer with PushingBox and Google Docs, but soon we realized that we could only get two sensor values working and we needed a better way of showing our data to our users. After doing some research we found the Blynk library that was simple to use and could display our data in a readable way. Our final GUI can be seen on Figure 16. This platform is also great if we would like to scale the number of nodes we wanted to display.

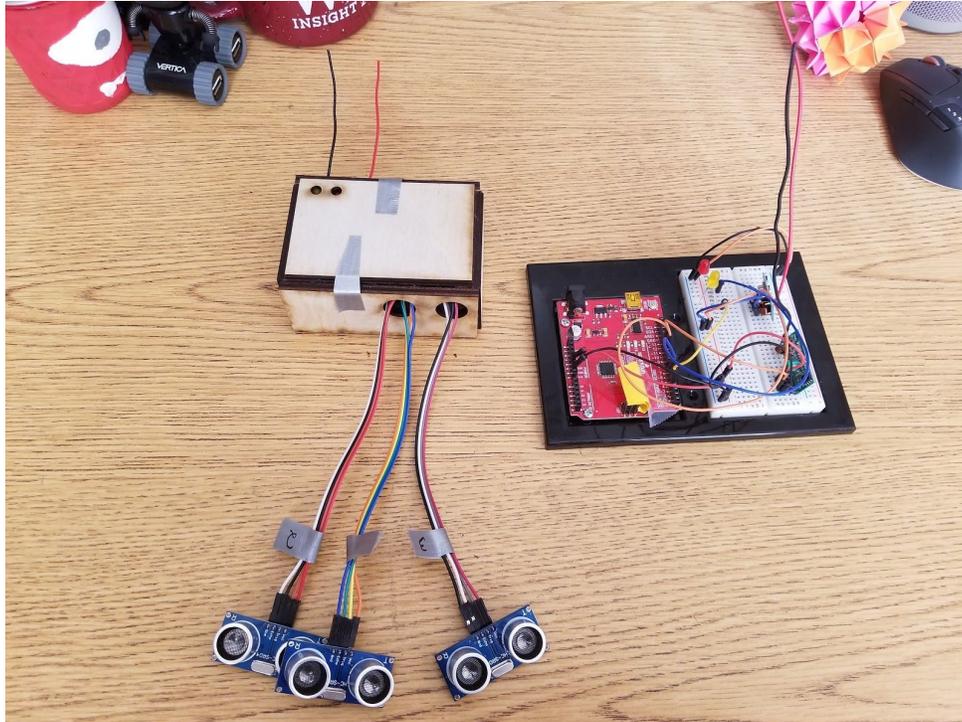


**Figure 16:** Web layer phone application GUI

The final iteration of the project is shown in Figure 17. The red light on the left node indicates that the sensor is sending messages, while the yellow light on the right node indicates that the sensor is listening for messages. During our experimentation we realized the transportation of the sensor nodes affected our time spent debugging the circuits because we would disconnect wires. We decided to fix this problem by laser cutting a box to enclose the whole circuit. This can be seen In Figure 18, the final product.



**Figure 17:** Final Outcome of Project



**Figure 18:** Final Product

## **Summary of Challenges Encountered and Lessons Learned**

The first and most important issue we had to deal with was understanding and implementing technical challenges with TDMA, especially relevant to the testing, debugging and packaging of data. After failing many times, we managed to have it working and transmitting data.

Another mistake we made was the fact that we soldered a 12.5 antenna to the modules while we should have used a 17 cm wire. We also had problems with the second iteration of the TDMA, possibly due to variability of sensors. We thought of removing the additional interference (“noise”) from the received message so that the receiver can authenticate the transmitting nodes. We reasoned that we could either implement a numerical filter or reset the variable.

We believe that by working on this project we understood and implemented a major and complex technology used in communications, the TDMA. It is quite complex since it requires precise synchronization between the transmitter and the receiver. We also acquired experience in designing a prototype where we realized the different phases of design and actual implementation. We feel the results of this project could contribute to a specific, identifiable need of the WPI academic community. We also feel that given more time and therefore, more improvement in its implementation, our prototype could be used in real-life situations, outside of our university. Finally, we also realized that technical challenges can occur in unpredictable ways sometimes, and when they occurred, we had to go back

and re-evaluate our strategy with the help of our TA. We believe that this enhanced our understanding of basic principles of communications and networks.

## **Discussion-Future Prospects**

The methodology followed in the development and deployment of a prototype application has included the following stages:

1. Breaking the concept in subsystems, designing the different subsystems of the prototype (Sensor Nodes, Aggregator, Web Interface)
2. Defining goals and constraints, creating a hierarchy of design requirements and extracting the overarching system architecture
3. Developing each subsystem as a set of modules (input/output, sensory/processing, action)
4. Integrating subsystems in a whole
5. Testing the functionality of the prototype
6. Usability testing

The resulting prototype should be considered at an early alpha stage in the software release alpha cycle (Reference: Producing Open Source Software - How to Run a Successful Free Software Project <https://producingoss.com/en/index.html> chapter 7)

This is a very early prototype that needs formal testing and further development. A method of evolutionary prototype may contribute to collect more feedback from the users and adapt the requirements analysis to the needs of the real users while advancing quickly in the path of normal system development (Reference: [http://www.cs.nott.ac.uk/~pszjg1/FSE12/FSE\\_7.pdf](http://www.cs.nott.ac.uk/~pszjg1/FSE12/FSE_7.pdf))

## **Conclusion**

Thankfully, our project worked! The web layer was launched successfully, and the nodes are communicating correctly with each other. However, every now and then, the nodes failed to communicate momentarily either due to network collisions, where two nodes were trying to transmit at the same time, or due to failure to send packets. We ended up implementing the essential idea of our proposal. We did not apply the idea of adding photoresistors, but we still ended up with an effective, elementary system using only ultrasonic sensors. One of the greatest takeaways from this project was that we learned a lot about TDMA and its implementation. Moreover, we learned more about how IoT works, by connecting physical devices through the network. Finally, our biggest problem was the hardware and especially the receivers. Nevertheless, this is acceptable due to their low cost.

## **References**

- Chesbrough, H. W. (2003). Open Innovation: The New Imperative for Creating And Profiting from Technology. Harvard Business Review Press.
- Christie, E.J., Jensen, D.D., Buckley, R.T., Menefee, D.A., Ziegler, K.K., Wood, K. and Crawford, R. (2012). Prototyping strategies: Literature review and identification of critical variables. ASEE Annual Conference and Exposition, Conference Proceedings
- Fujimoto, S. T. (2000). The Effect of “Front-Loading” Problem-Solving on Product. *Journal of Product Innovation Management*, 17, 128-142.
- Koen, P.A., Bertels, H. M. J. and Kleinschmidt E.J.( May-June, 2014). Managing the Front End of Innovation—Part II. *Research-Technology Management*
- Li, Y. and Bartos, R. A survey of protocols for Intermittently Connected Delay-Tolerant Wireless Sensor Networks (2014) *Journal of Network and Computer Applications* 41: 411-423
- Miao, G., J. Zander, J., K.W. Sung, K.W. and Slimane, B. (2016). *Fundamentals of Mobile Data Networks*. Cambridge University Press. ISBN 1107143217.
- Peisert, S., Talbot, E., and Kroeger, T. (2013) Principles of Authentication, in *Proceedings of the 2013 New Security Paradigms Workshop*, pp. 47-56, available at <https://dl.acm.org/citation.cfm?doid=2535813.2535819>
- Terho H., et al (2017) Understanding the Relations Between Iterative Cycles in Software Engineering, paper presented at the 50th Hawaii International Conference on System Sciences, available at [http://aisel.aisnet.org/hicss-50/st/agile\\_development/7/](http://aisel.aisnet.org/hicss-50/st/agile_development/7/)
- Wenger-Trayner, E. A. (April 15, 2015). Communities of practice – a brief introduction. Wenger-Trayner. Retrieved from <http://wenger-trayner.com/introduction-to-communities-of-practice/>
- Yang , S-H. (2014) *Wireless Sensor Networks Principles, Design and Applications*, Springer, available at [ftp://doc.nit.ac.ir/cee/m.zahabi/BOOKS/wsn%20book/Shuang-Hua%20Yang%20\(aut.h.\)-Wireless%20Sensor%20Networks\\_%20Principles,%20Design%20and%20Applications-Springer%20London%20\(2014\).pdf](ftp://doc.nit.ac.ir/cee/m.zahabi/BOOKS/wsn%20book/Shuang-Hua%20Yang%20(aut.h.)-Wireless%20Sensor%20Networks_%20Principles,%20Design%20and%20Applications-Springer%20London%20(2014).pdf)

## **Datasheets**

1. Arduino Nano V3:
  1. <http://roboromania.ro/datasheet/Arduino-Nano-roboromania.pdf> 2.
- Ultrasonic Sensors:
  1. <http://www.micropik.com/PDF/HCSR04.pdf>
  2. [https://elec Freaks.com/estore/download/EF03085-HC-SR04\\_Ultrasonic\\_Module\\_User\\_Guide.pdf](https://elec Freaks.com/estore/download/EF03085-HC-SR04_Ultrasonic_Module_User_Guide.pdf)
  3. With LCD- <http://raspoid.com/download/datasheet/HCSR04>
3. 433MHz RF front end RX and TX modules
  1. <http://arduino basics.blogspot.com/2014/06/433-mhz-rf-module-with-arduino-tutorial.html>
  2. [https://www.pjrc.com/teensy/td\\_libs\\_VirtualWire.html](https://www.pjrc.com/teensy/td_libs_VirtualWire.html)
  3. <https://www.youtube.com/watch?v=b5C9SPVIU4U&t=1092s>
4. LCD
  1. <https://www.arduino.cc/en/Tutorial/LiquidCrystalDisplay>

## **Appendix**

### **Design proposal Document**

#### **Abstract**

One of the most visited places on Campus is the Gordon Library which offers space to students that wish to find a quiet place to study. However, at peak times, many students have a hard time finding a seat in the cubicle desk spaces located in the library. Efficient use of time is essential for successful accomplishment of course assignments. We decided to address this need of the WPI community by designing and implementing a wireless network capable of connecting multiple sensors to the Internet in order to obtain information about which desk cubicles are empty, therefore available to use, at any particular time. Each cubicle will possess a simple sensor and an embedded processor detecting whether the space is occupied or not. This information will be transmitted via the Internet to the members of the WPI community seeking an empty cubicle space to work.

#### **Introduction to the problem statement**

Have you ever been to the library on a busy week day and have not found where to study? How about in finals week when there is no time to lose? Or have you ever been to the Campus Center or even the Dining Hall during lunch or dinner and found it to be packed? Yes, these places are highly sought around campus, especially during the busy weeks of the term. As the population of both undergraduate and graduate students on campus increases, demand for space on campus increases as well. Usually, most students seek places around campus to study and focus on work, but it is getting harder to find them fast, sometimes taking up to 20 minutes or more. This is a great problem for WPI students' tight schedules. There has to be a better way to let the WPI student population know if they should make a run to the Gordon Library to successfully find a place to study or to the Campus Center to get something to eat. Efficient use of time in a busy and demanding academic environment is essential for successful completion of work requirements, for students and faculty alike. Especially for students, any measure to facilitate better organization of their time is indispensable. For all these reasons we decided to design a system by which any member of the WPI community may know at any given time whether the particular place/environment they are looking for, is crowded or empty, and therefore, make an informed decision about its availability and use. Our project will focus on identifying empty cubicle desk spaces in the Gordon Library and communicating it to the WPI community via the Internet.

#### **Description of proposed solution (including rationale)**

Our objective is to design and implement an IoT system, which comprises a wireless network capable of connecting multiple low complexity devices (sensors) to the Internet in order to obtain information about which Gordon Library cubicles are empty, therefore available to use, at any particular time. Each cubicle will possess a simple sensor connected to an embedded processor. The desired task will be to detect whether the cubicle is occupied

or not. This information will be transmitted via the Internet to the members of the WPI community seeking an empty cubicle space to work.

### Brief overview of implementation

The first step in the implementation process of our IoT system will be the Physical Layer. For the processing of data we will be using the Arduino Nano V3.0 Microcontroller and Arduino Uno. This cost effective microcontroller has small dimensions and possesses the input and output pins for our system. The main purpose of this microcontroller will be to interface with the Ultrasonic sensor we have decided on using. Furthermore, we will be able to add more modules to our system as we see fit because of the microcontrollers' array of ports to choose from (shown in Figure 1).

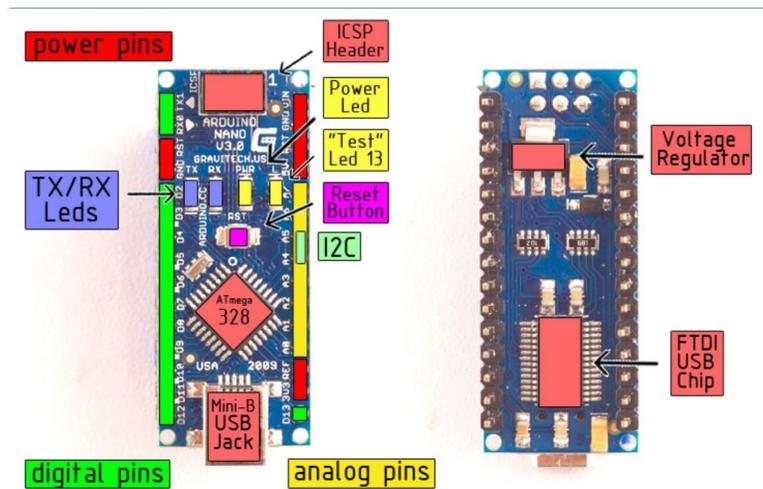


Figure 1. Arduino Nano V3.0 Microcontroller.

As mentioned above, the microcontroller will interface with the HC-SR04 Ultrasonic Sensor/s (shown in Figure 2) in order to monitor the status of a cubicle. The Ultrasonic Sensor operates at a frequency of 40KHz, which is beyond the human hearing capacity. In addition, it has a range from 2 cm-4 m which is perfect for the purpose of our system. The Ultrasonic sensor will be placed directly above the cubicle to be monitored and report its status every T seconds. The Echo and Trigger pins are the main pins we will be using in the Ultrasonic Sensor. By sending a 10us pulse (high state) to the trigger pin the module will send out an 8 cycle sonic burst. Then, the echo pin will receive this burst and output the time in microseconds. This communication protocol is referred as TTL (Transistor-Transistor Logic) which means that the level will always be logic high or logic low (a square wave). Therefore, we can use this to find the distance the object (human) is from the Ultrasonic Sensor with the equation shown below.

$$distance = duration * 0.034$$

Figure 2. HC-SR04 Ultrasonic Sensor Distance Equation

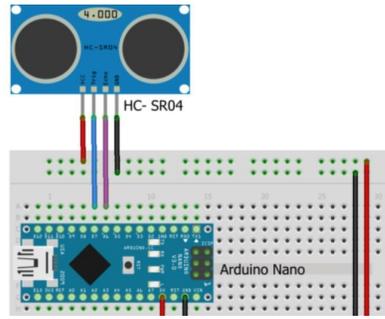


Figure 3. HC-SR04 Ultrasonic Sensor

In order to connect our sensor nodes to our aggregator node we will use the 433MHz RF front end RX and TX modules. Apart from being inexpensive, they are really easy to use and make our system have a minimalistic design. The Transmitter possesses three pins: GND, Vcc and DATA. We will be using the DATA pin that is connected to a Digital pin on the arduino to transmit the signal, in this case the sensor data, to our Receiver. On the other hand, the Receiver possesses four pins: GND, Vcc and two DATA pins. We will be using only one DATA pin (shown in Figure 3). According to our research, adding an antenna to the Tx and Rx modules would enable the signal of our system to be propagated at greater distances, therefore solving any attenuation/path loss that the signal could encounter. However, it was brought to our attention that over amplifying the signal could cause interference with other signals and could possibly get us in trouble with authorities, therefore we will be careful to keep the signal within range and consult with the professor when in doubt. Apart from that, we also realized from various resources that we had to test the distance ourselves with the various antenna configurations for better results. Ideally, we will use an antenna of half or a quarter of the wavelength (17.3 cm) because these modules transmit at a frequency of 433 MHz with a wavelength of 69.24 cm. We will start by adding 17.3 cm of coiled up antenna to the receiver and 25 cm of straight antenna to the transmitter. This should give us a pretty good range to start with. Furthermore, these modules use an ASK (Amplitude Shift Keying) which modulates the carrier wave and can be easily encoded and decoded, but is limited to the amount of data it can send. To deal with noise, we will use the “radio head” library that takes care of keying to reduce noise and provides us with an array of functions to transmit/receive data.

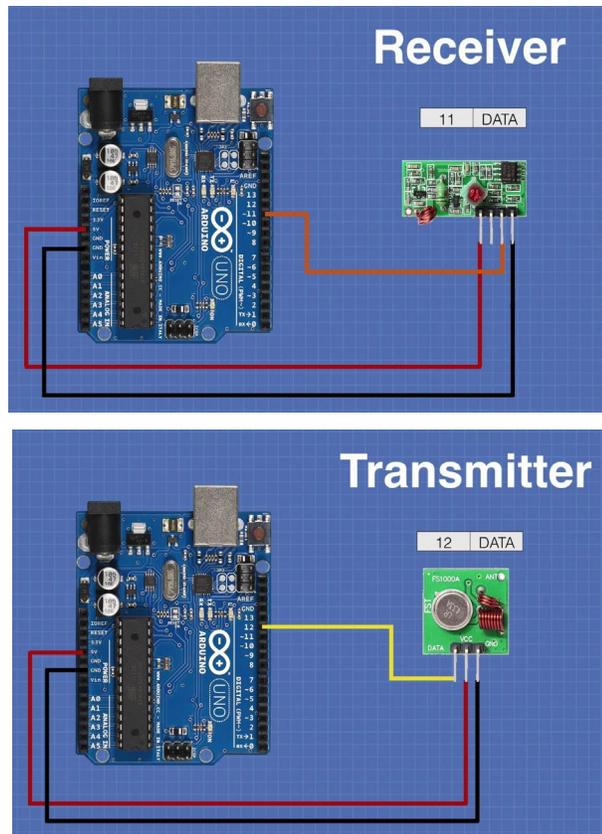


Figure 4. Data pin on Receiver (left) and Transmitter (right).

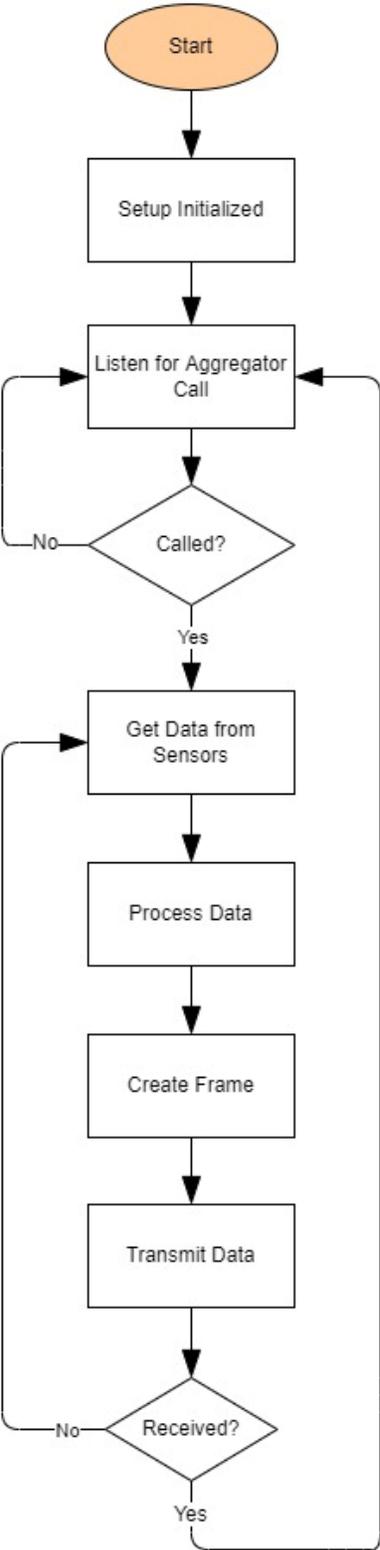
The final components we will add to the physical layer are a LCD and LEDs. For the LCD we will use the “Crystal library” to show the status of the modules and whether data is being received. The LEDs will serve as indicators that the modules are working correctly. These two components will make debugging the system a lot easier. Our team may decide to add more sensors to the system as we see fit.

Estimated Parts List With Individual Costs		
Name	Quantity	Cost per unit
Ultrasonic Module HC-SR04 Distance Sensor	TB D	\$2.00
Arduino Nano V3.0, Elegoo Nano board CH340/ATmega328P	3	\$4.29

UNO R3 Board ATmega328P ATMEGA16U2	1	\$10.90
Gowoops 433Mhz RF Transmitter and Receiver	5 pai rs	\$1.80
LCD Display	1	\$3.90
LED Lights	TB D	\$0.04

The second step in the implementation process is the Medium Access Control Layer. This will be mainly implemented in Arduino and MatLab code. We will have to keep track of which sensor is sending data and from what area of the library (microprocessor) it is being sent. This can be done by framing a datagram and adding overhead that specifies the sensor number and the embedded processor name. This packet will then be sent to the aggregator node where the datagram will be filtered through the right channels to be executed. We will be able to correlate data to a specific sensor and embedded processor that will allow us to implement a transport layer to test the reliability of the data received. This will also allow our user interface to distinguish how to display the data. Furthermore, there are two possible ways of how we will address handling multiple sensor nodes that communicate with one aggregator node. The first possible implementation is the Time Division Multiple Access (TDMA) approach. This multiple channel access method will allow the system to communicate over one frequency (433 MHz). Time will be divided into slots and only one sensor node will be able to transmit during each slot. For this case the aggregator will serve as a master that calls (transmits) the sensor node name that it wants to communicate with. Then the sensor node will send out the frame to the aggregator and once it finishes processing the data, the aggregator node will call on the next sensor node it wants to communicate with. This will require that our sensor nodes and aggregator node have a pair or transmitter and receiver modules for each node. Additionally, the system will be able to check for corrupted files and data that is not received with a parity checking process. On the other hand, the second multiple channel access method we could implement is the Code Division Multiple Accesses (CDMA). This method will add complexity to the code implementation of the aggregator node. The aggregator node will have to filter out all the sensors nodes that are transmitting and focus on the one it is listening for. The sensor nodes will therefore be transmitting continuously and could be accessed at any time. The reason behind this, is that ideally in the physical layer of our system, this will allow us to use only transmitters on the sensor nodes and one receiver on the aggregator node. This will be able to have an error/corruption checking scheme, namely the CRC scheme, but unfortunately it cannot re-transmit data because of the one transmitter-one receiver implementation. Therefore, that data will be lost.

### Sensor Node Flowchart



### Aggregator Node Flowchart

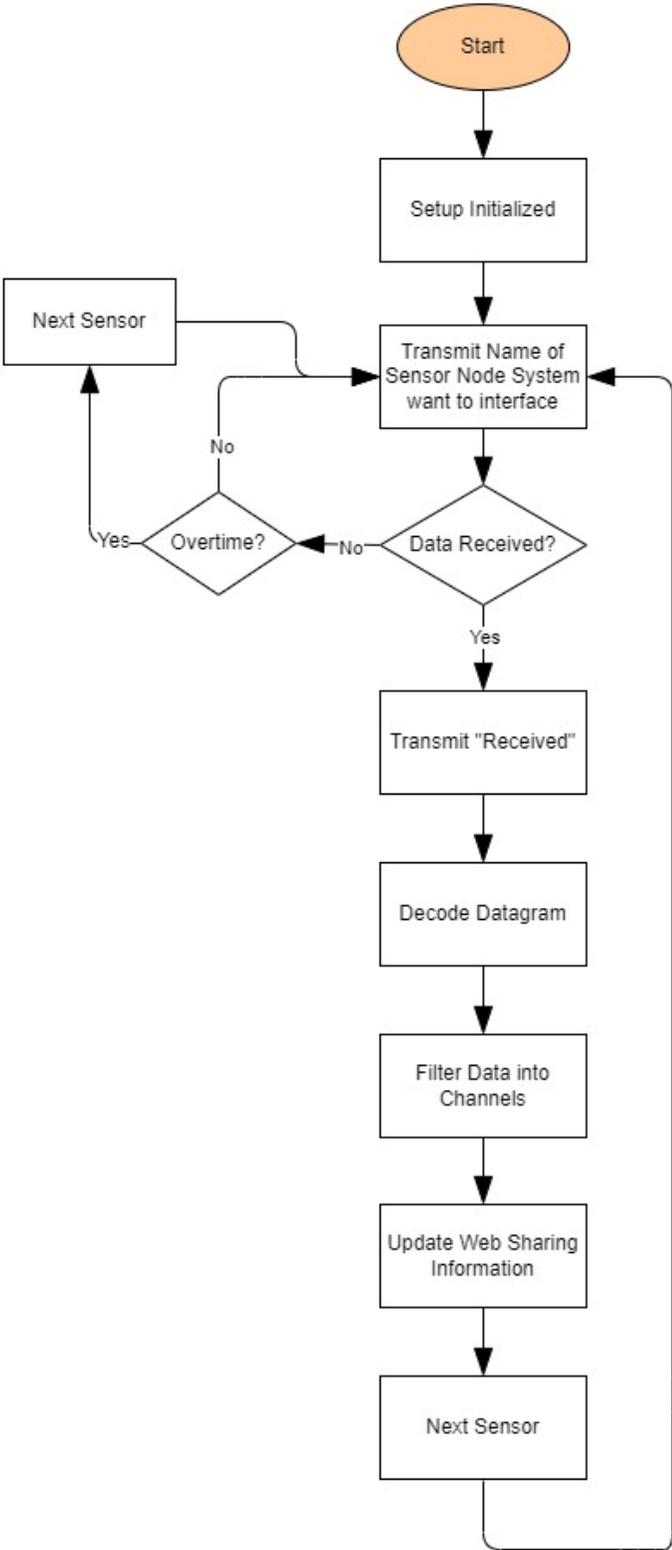


Figure 5. Sensor Node (left) and Aggregator Node (right) Activity Diagrams.

The third and fourth step of the implementation process are somewhat connected, the layers being the connectivity to the Internet and the Web Sharing Information. For the Connectivity to the Internet Layer we will be using the Blynk Library with a unique authentication token in order to interface between the arduino and the application on our phones. This will allow us to take that data from the Applications layer of the aggregator node and send it across the web to a user. This brings us to the Web Sharing of Information Layer where we will use the Blynk Application as mentioned before. This application will allow us to display data from the sensors across the internet with an easy-to-read graphical user interface (GUI). Some of the features that this GUI will display are the status of seats and cubicles in different areas of the library. Another program that our system could interface with is “ThinkSpeak MATLAB”. Our team will further implement this section of the project as we see fit for our system.

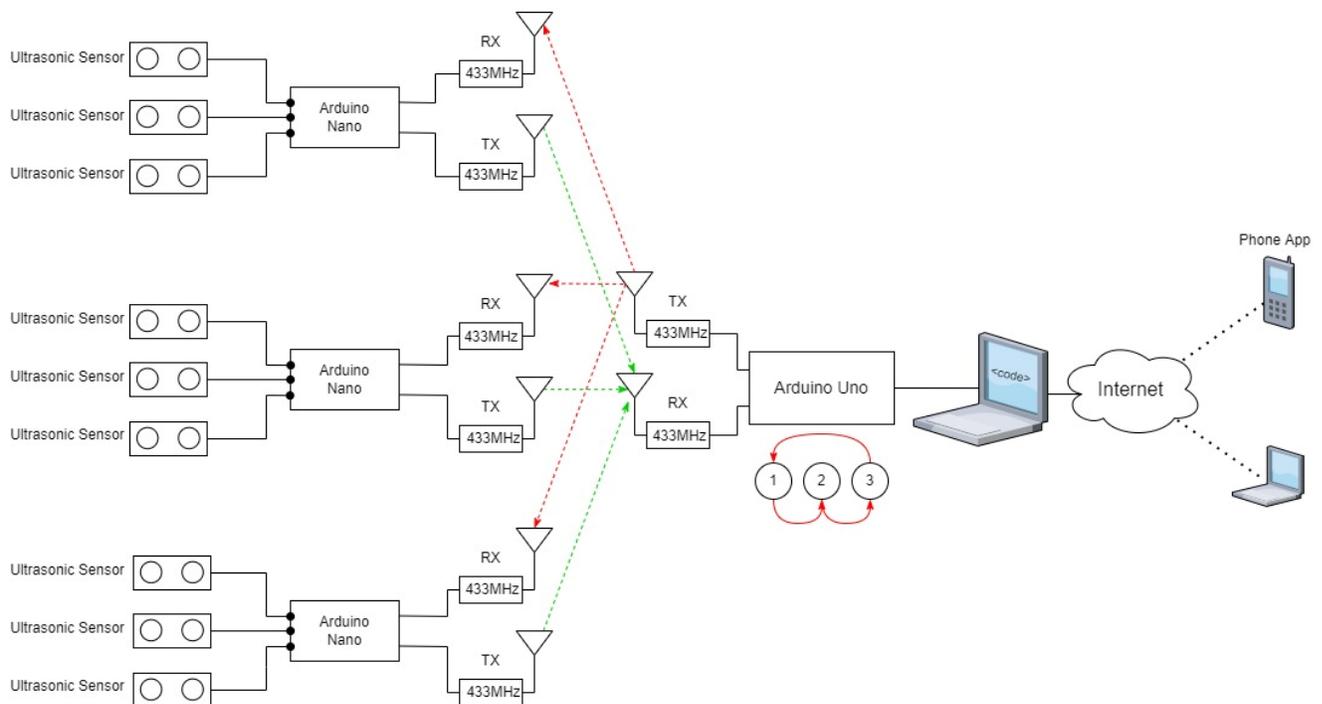


Figure 6. Illustration of the IoT architecture for a wireless sensor network used to detect the presence of humans.

### Prototype development methodology and evaluation strategies

We will use the following prototyping methodology:

1. Break the concept in subsystems, design the different subsystems of the prototype (Sensor Nodes, Aggregator, Web Interface)
2. Define goals and constraints, create a hierarchy of design requirements and extract the overarching system architecture

3. Develop each subsystem as a set of modules (input/output, sensory/processing, action)
4. Integrate subsystems in a whole
5. Test the functionality of the prototype
6. Usability testing

We will implement the following steps for the completion and evaluation of our prototype

1. We will make sure the prototype is correct and faithful to the design. As we will build the prototype, we will make the necessary changes and fix errors that may come up. In this way we will employ a strategy that identifies and solves problems in early phases of the development of our prototype (otherwise called “front loading” problem solving).
2. We will conduct multiple rounds of usability testing with increasing levels of fidelity.
3. We will consider the limitations of our prototype, thus defining what we can and cannot test.
4. We will create a “community of practice” (CoP) which would be a group of people from the WPI community (undergraduate and graduate students, faculty, administrators) that share the same need of solving the problem we are attempting to solve, i.e. the need of knowing which common spaces on campus are available to use.. We will explain our prototype to them and ask them to use it and share their views and comments on its usability. From the literature it is clear that employment of CoPs in prototype evaluation strongly correlates with innovation.
5. We will create a video that we will upload to YouTube, in order to showcase our prototype.

**Project logistics (e.g., timeline, milestones, task specification)**

TIMELINE	MILESTONES AND TASK SPECIFICATION
March 30	<ul style="list-style-type: none"><li>• Complete Proposal Document</li></ul>
April 6	<ul style="list-style-type: none"><li>• Setup client and complete functionality of individual board without networking</li></ul>
April 13	<ul style="list-style-type: none"><li>• Set up local communication between clients and “server”</li><li>• Initial setup for web-interface</li></ul>
April 19	<ul style="list-style-type: none"><li>• Complete web interface</li></ul>
April 27	<ul style="list-style-type: none"><li>• Fix bugs</li></ul>
May 1	<ul style="list-style-type: none"><li>• Final Report</li></ul>

**Source Code & video**

The source code and video can be found in the following link:



[https://github.com/FivosJKavassalis/ECE2305\\_Team14](https://github.com/FivosJKavassalis/ECE2305_Team14)



<https://www.youtube.com/watch?v=R3chP828VOM&t=15s>