



---

# CONTEST 2022

---

SMART HOME  
AUTOMATION USING  
W5500-EVB-Pico

By Albin Joseph

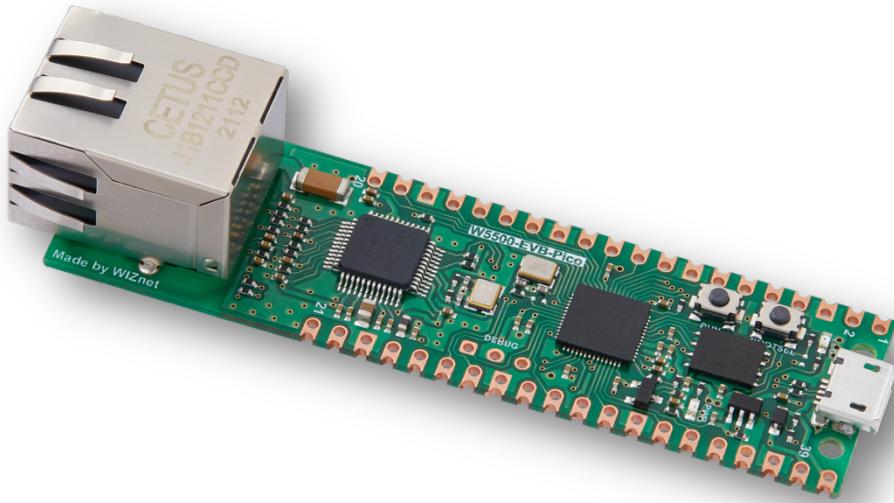
## **Contents:-**

- 1) Quick Intro to W5500-EVB-Pico**
- 2) Intro to Project / Aim of the Project**
- 3) Block Diagram of the Project**
- 4) Hardware and Material Used**
- 5) Problems Challenges and Solutions**
- 6) R&D**
- 7) Flow Diagram for Alexa and Google Assistant  
with Developer Console Images**
- 8) Sample Shortcuts Code and Shortcuts  
Achievement**
- 9) Circuit Python Code and Adafruit IO Dashboard**
- 10) Results and Conclusion**

# **Problem Statement**

To convert normal home devices to smart home devices by creating an automated environment at a cheaper cost by making use of W5100s-evb-pico with internet connectivity and by proving it with a mini home demo model to create a similar environment as that of an actual home.

# 1) Quick Intro to W5500-EVB-Pico



W5500-EVB-Pico is a microcontroller evaluation board based on the Raspberry Pi RP2040 and fully hardwired TCP/IP controller W5500 – and basically works the same as Raspberry Pi Pico board but with additional Ethernet via W5500.

- Raspberry Pi Pico Clone
- Ethernet (W5500 Hardwired TCP/IP CHIP)

It is one of the best things on the planet from Wiznet!

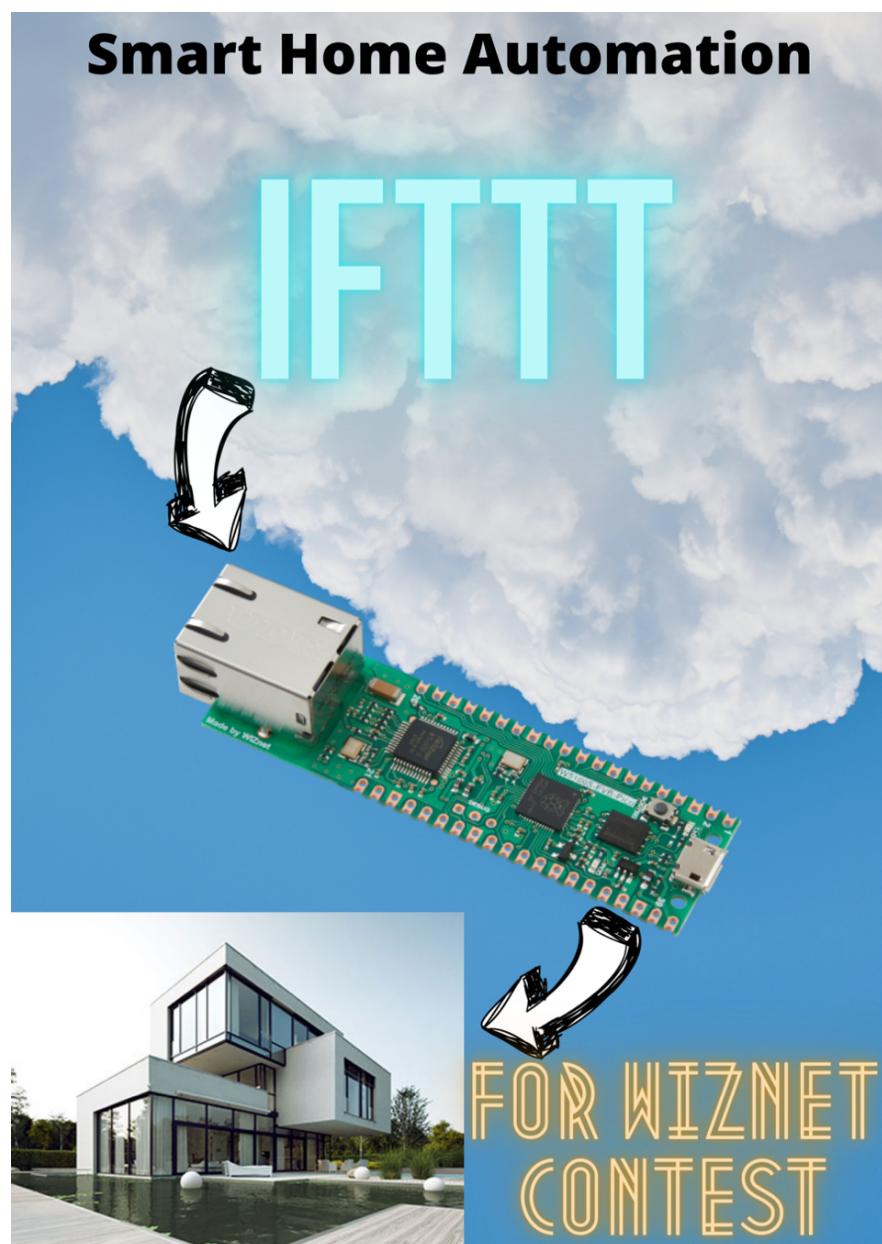
More details can be found I the following link :-

<https://docs.wiznet.io/Product/iEthernet/W5500/w5500-evb-pico/>

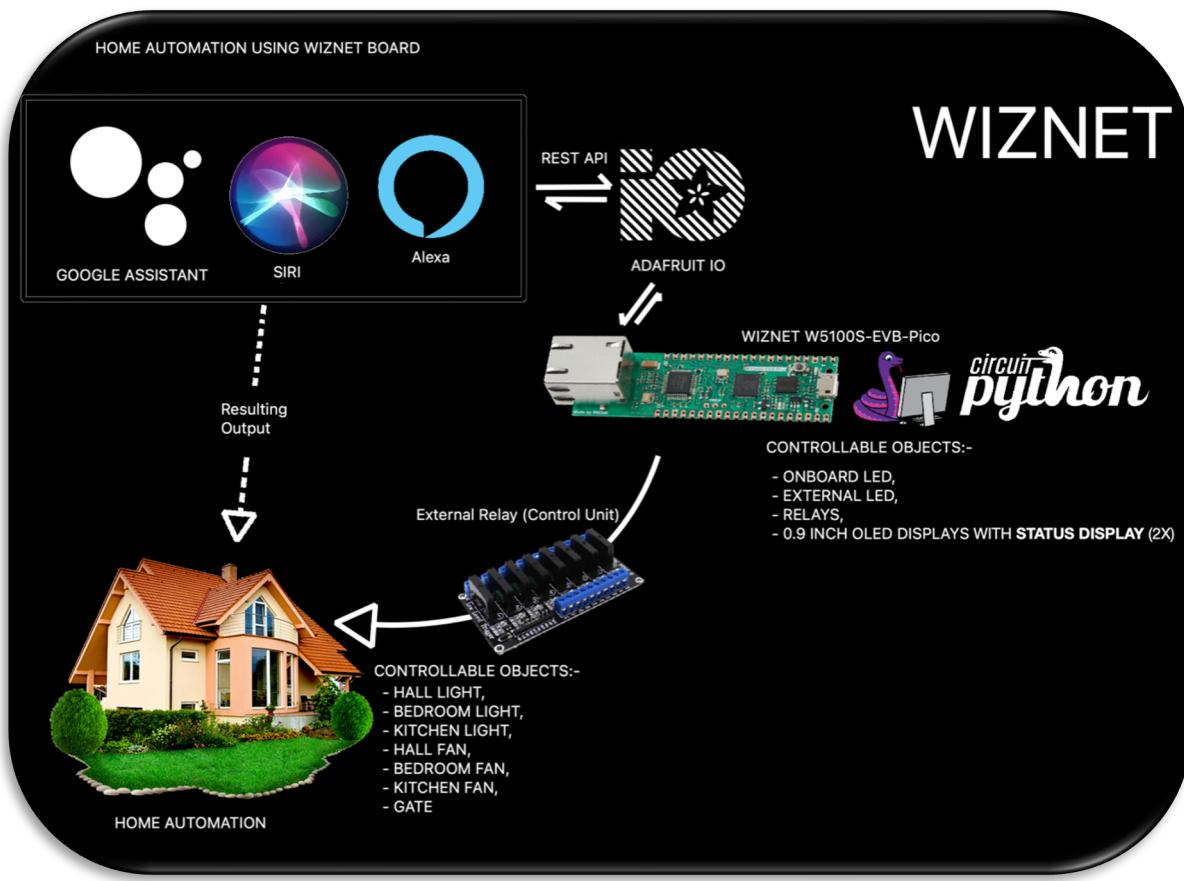
## 2) Intro to Project / Aim of the Project

The Aim of the project is to turn normal home devices to smart devices by creating an automated environment with the help of W5100s-evb-pico with internet connectivity to create a low cost home automation setup.

The Below was the proposed project flyer.... But things changed... changed for good... you will find out.



### 3) Block Diagram of the Project



The above Block Diagram shows the interaction flow (behind the scenes) of what actually happens when you interact with your favourite assistant !

## 4) Hardware and Material Used

	W5500-EVB-Pico -----	x 1
	0.9 Inch OLED Display -----	x2
	Bread Board -----	x1
	8 Channel Relay Module -----	x1
	Mini fans -----	x2
	DC Motor and Fan Blade -----	x1
	0.5μF capacitor for DC Motor -----	x1
	Acrylic Sheets 3 types -----	x5
	Plywood 2 types -----	x2
	False Ceiling Lights -----	x3
	Old CD/DVD Drive -----	x1

## 5) Problems, Challenges and Solutions

There were many challenges faced starting from gathering the right parts but the main issue that was face was when the I tried to use a DC Motor fan, somehow the back EMF and magnetic interference was causing the relay module to stop functioning as soon as the motor starts and because the relay malfunctioned it was resetting the circuit python program and used to go on this reset loop finally causing the board to freeze and I had to end up nuking the board and re-flashing circuit python and the code.

Well actually there seemed two ways to solve this problem and the first and easy way was to replace the dc motor (which is used as hall fan) to a tiny propeller which would instantly solve the problem, but instead I went the tough way to solve this problem and started to learn about the reason behind it and solve it, it took me almost 3 weeks to find the solution to this, and at the end I was able to solve it using a  $0.5\mu\text{F}$  capacitor which was connected in parallel to the DC motor along with a 1 ohm resistor, which did the job, but also a varistor could have been used too to solve this which unfortunately seemed to be very rare to find.

The other problem faced was to stick the Acrylic sheets together the available solvents and glue was not at all strong enough to hold it in place so I had to research and develop a solvent along with my father, the solvent was made by dropping acrylic pieces into Chloroform and using that solution to weld it together after cleaning up with alcohol The Acrylic sticks together due to chemical bonding.

## 6) R&D

First thing to start off with the project was to create an actual house model and it was almost impossible to join the acrylic sheets together it seemed to fall off one by one causing a tiny domino effect. Somehow I had to figure out a way to fix those sheets together so after a bit of research me along with my dad was able to drop some acrylic sheets into chloroform to prepare its on solvent which made it bond at a molecular level.

The other part where the R&D was done was on the software side... well I could have made use of IFTTT (Short for IF This Then That) a 3<sup>rd</sup> party service that can be used to connect Adafruit IO's feeds with Google Assistant or Alexa and can also be used to connect Siri with Adafruit IO's feeds by making use of webhooks provided by IFTTT which seemed easy and pretty straight forward right? And indeed it was my first plan and that is why you did see IFTTT on my initial project proposal flyer.

I was also wondering if I could just make Api calls and directly interact with Adafruit IO feeds as I was graving to know how it worked and I started to feel it to myself, "What if I could get rid of IFTTT, after all it is just a 3<sup>rd</sup> party" and so I ended up in this Adafruit IO's Api page which is located at

<https://io.adafruit.com/api/docs/#adafruit-io-http-api> but most of it just showed how to fetch the data from in and not how to make use of it to trigger a feed, so after a few

trial and error I was able to find the right method to send the value to trigger the feed using curl in terminal which led a way to completely let go of IFTTT and send out POST request to actually trigger the Adafruit Io's feed, it has also made it possible for me to later found a way to integrate it into Siri without using webhooks and directly sending requests to Adafruit IO's feeds and I can bet you can never find how it is done on shortcuts over the internet as I did search for the same across many forms, articles, YouTube videos and tutorial points they all just seemed know how to connect shortcuts to Adafruit IO's feed only with IFTTT. more of these details can be found in module 10 of this document.

## 7) Flow Diagram for Alexa and Google Assistant



## Alexa Developer console :-

The screenshot shows the Alexa Developer Console interface. The top navigation bar includes links for 'Your Skills', 'AlexaSkills Contest 2022', 'Build', 'Code', 'Test', 'Distribution', 'Certification', and 'Analytics'. On the far right are search and user profile icons. The left sidebar has a 'CUSTOM' section with 'Invocations', 'Interaction Model', 'Assets' (which includes 'Slot Types (0)', 'Multimodal Responses', 'Interfaces', and 'Endpoint'), and 'MODELS' and 'TOOLS' sections. The main content area features a large banner for the 'Alexa Skills Kit Developer Tutorial for Programmers: Building with the Alexa Developer Console'. Below the banner, there's a 'Watch on YouTube' button. The 'Resources' section contains links for 'Technical Walkthrough: Toolkit for VS Code', 'Watch: Simulate Skills in VS Code with Alexa Skills Toolkit', 'Make Money', 'Feature Updates & Releases', 'Getting Started: A Comprehensive Course (Cake Time)', and 'Errors and Warnings'. The 'Skill builder checklist' on the right lists five steps: 1. Invocation Name (with a note to enter an invocation name), 2. Intents, Samples, and Slots (with a note to add at least one intent and one sample utterance), 3. Build Model (with a note to successfully build your interaction model), 4. Endpoint (with a note to set a web service endpoint to handle skill requests), and 5. Monetize Your Skill (with a note to add In-Skill Products or offer a Paid Skill). Each step has a green checkmark icon.

The screenshot shows the Alexa developer console interface. At the top, there's a navigation bar with 'Your Skills', 'Alexa Skills Contest 2022', 'Build', 'Code', 'Test', 'Distribution', 'Certification', and 'Analytics'. Below the navigation bar is a toolbar with icons for 'File', 'Edit', 'Delete', 'Rename', 'Database', 'AWS Lambda', 'Storage', 'CloudWatch Metrics', 'Logs', 'Latency', 'Integrate', 'Deploy', 'Import Code', 'Offline Tools', and 'Help'. On the right side, there are buttons for 'Save', 'Deploy', and 'Promote to live'. A status message at the bottom right says 'Deploy before testing your skill'.

The main area is a code editor titled 'local-debugger.js'. The code is a Node.js script for an Alexa skill. It includes handlers for various intents like 'AMAZON.CancelIntent', 'AMAZON.StopIntent', 'AMAZON.FallbackIntent', 'IntentRequest', 'SessionEndRequest', and 'IntentReflecterHandler'. It also includes error handling for 'Error' and 'ErrorBoundary'. The code uses the Alexa Skills Kit (ASK) framework.

```
alex developer console
Your Skills Alexa Skills Contest 2022 Build Code Test Distribution Certification Analytics

File Edit Delete Rename Database AWS Lambda Storage CloudWatch Metrics Logs Latency Integrate Deploy Import Code Offline Tools Help

local-debugger.js
local-debugger.js
index.js x
local-debugger.js
util.js

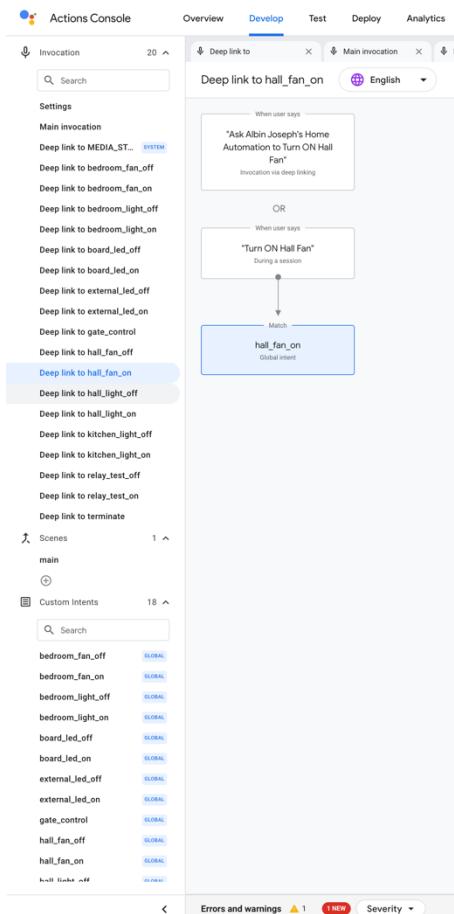
local-debugger.js
index.js x
local-debugger.js
util.js

1  // (Optional) get IntentNameHandlerInput.requestEnvelope --> 'AMAZON.CancelIntent'
2  2  // (Optional) get IntentNameHandlerInput.requestEnvelope --> 'AMAZON.StopIntent';
3  3
4  4
5  5
6  6
7  7
8  8
9  9
10 10
11 11
12 12
13 13
14 14
15 15
16 16
17 17
18 18
19 19
20 20
21 21
22 22
23 23
24 24
25 25
26 26
27 27
28 28
29 29
30 30
31 31
32 32
33 33
34 34
35 35
36 36
37 37
38 38
39 39
40 40
41 41
42 42
43 43
44 44
45 45
46 46
47 47
48 48
49 49
50 50
51 51
52 52
53 53
54 54
55 55
56 56
57 57
58 58
59 59
60 60
61 61
62 62
63 63
64 64
65 65
66 66
67 67
68 68
69 69
70 70
71 71
72 72
73 73
74 74
75 75
76 76
77 77
78 78
79 79
80 80
81 81
82 82
83 83
84 84
85 85
86 86
87 87
88 88
89 89
90 90
91 91
92 92
93 93
94 94
95 95
96 96
97 97
98 98
99 99
100 100
101 101
102 102
103 103
104 104
105 105
106 106
107 107
108 108
109 109
110 110
111 111
112 112
113 113
114 114
115 115
116 116
117 117
118 118
119 119
120 120
121 121
122 122
123 123
124 124
125 125
126 126
127 127
128 128
129 129
130 130
131 131
132 132
133 133
134 134
135 135
136 136
137 137
138 138
139 139
140 140
141 141
142 142
143 143
144 144
145 145
146 146
147 147
148 148
149 149
150 150
151 151
152 152
153 153
154 154
155 155
156 156
157 157
158 158
159 159
160 160
161 161
162 162
163 163
164 164
165 165
166 166
167 167
168 168
169 169
170 170
171 171
172 172
173 173
174 174
175 175
176 176
177 177
178 178
179 179
180 180
181 181
182 182
183 183
184 184
185 185
186 186
187 187
188 188
189 189
190 190
191 191
192 192
193 193
194 194
195 195
196 196
197 197
198 198
199 199
200 200
201 201
202 202
203 203
204 204
205 205
206 206
207 207
208 208
209 209
210 210
211 211
212 212
213 213
214 214
215 215
216 216
217 217
218 218
219 219
220 220
221 221
222 222
223 223
224 224
225 225
226 226
227 227
228 228
229 229
230 230
231 231
232 232
233 233
234 234
235 235
236 236
237 237
238 238
239 239
240 240
241 241
242 242
243 243
244 244
245 245
246 246
247 247
248 248
249 249
250 250
251 251
252 252
253 253
254 254
255 255
256 256
257 257
258 258
259 259
260 260
261 261
262 262
263 263
264 264
265 265
266 266
267 267
268 268
269 269
270 270
271 271
272 272
273 273
274 274
275 275
276 276
277 277
278 278
279 279
280 280
281 281
282 282
283 283
284 284
285 285
286 286
287 287
288 288
289 289
290 290
291 291
292 292
293 293
294 294
295 295
296 296
297 297
298 298
299 299
300 300
301 301
302 302
303 303
304 304
305 305
306 306
307 307
308 308
309 309
310 310
311 311
312 312
313 313
314 314
315 315
316 316
317 317
318 318
319 319
320 320
321 321
322 322
323 323
324 324
325 325
326 326
327 327
328 328
329 329
330 330
331 331
332 332
333 333
334 334
335 335
336 336
337 337
338 338
339 339
340 340
341 341
342 342
343 343
344 344
345 345
346 346
347 347
348 348
349 349
350 350
351 351
352 352
353 353
354 354
355 355
356 356
357 357
358 358
359 359
360 360
361 361
362 362
363 363
364 364
365 365
366 366
367 367
368 368
369 369
370 370
371 371
372 372
373 373
374 374
375 375
376 376
377 377
378 378
379 379
380 380
381 381
382 382
383 383
384 384
385 385
386 386
387 387
388 388
389 389
390 390
391 391
392 392
393 393
394 394
395 395
396 396
397 397
398 398
399 399
400 400
401 401
402 402
403 403
404 404
405 405
406 406
407 407
408 408
409 409
410 410
411 411
412 412
413 413
414 414
415 415
416 416
417 417
418 418
419 419
420 420
421 421
422 422
423 423
424 424
425 425
426 426
427 427
428 428
429 429
430 430
431 431
432 432
433 433
434 434
435 435
436 436
437 437
438 438
439 439
440 440
441 441
442 442
443 443
444 444
445 445
446 446
447 447
448 448
449 449
450 450
451 451
452 452
453 453
454 454
455 455
456 456
457 457
458 458
459 459
460 460
461 461
462 462
463 463
464 464
465 465
466 466
467 467
468 468
469 469
470 470
471 471
472 472
473 473
474 474
475 475
476 476
477 477
478 478
479 479
480 480
481 481
482 482
483 483
484 484
485 485
486 486
487 487
488 488
489 489
490 490
491 491
492 492
493 493
494 494
495 495
496 496
497 497
498 498
499 499
500 500
501 501
502 502
503 503
504 504
505 505
506 506
507 507
508 508
509 509
510 510
511 511
512 512
513 513
514 514
515 515
516 516
517 517
518 518
519 519
520 520
521 521
522 522
523 523
524 524
525 525
526 526
527 527
528 528
529 529
530 530
531 531
532 532
533 533
534 534
535 535
536 536
537 537
538 538
539 539
540 540
541 541
542 542
543 543
544 544
545 545
546 546
547 547
548 548
549 549
550 550
551 551
552 552
553 553
554 554
555 555
556 556
557 557
558 558
559 559
560 560
561 561
562 562
563 563
564 564
565 565
566 566
567 567
568 568
569 569
570 570
571 571
572 572
573 573
574 574
575 575
576 576
577 577
578 578
579 579
580 580
581 581
582 582
583 583
584 584
585 585
586 586
587 587
588 588
589 589
590 590
591 591
592 592
593 593
594 594
595 595
596 596
597 597
598 598
599 599
600 600
601 601
602 602
603 603
604 604
605 605
606 606
607 607
608 608
609 609
610 610
611 611
612 612
613 613
614 614
615 615
616 616
617 617
618 618
619 619
620 620
621 621
622 622
623 623
624 624
625 625
626 626
627 627
628 628
629 629
630 630
631 631
632 632
633 633
634 634
635 635
636 636
637 637
638 638
639 639
640 640
641 641
642 642
643 643
644 644
645 645
646 646
647 647
648 648
649 649
650 650
651 651
652 652
653 653
654 654
655 655
656 656
657 657
658 658
659 659
660 660
661 661
662 662
663 663
664 664
665 665
666 666
667 667
668 668
669 669
670 670
671 671
672 672
673 673
674 674
675 675
676 676
677 677
678 678
679 679
680 680
681 681
682 682
683 683
684 684
685 685
686 686
687 687
688 688
689 689
690 690
691 691
692 692
693 693
694 694
695 695
696 696
697 697
698 698
699 699
700 700
701 701
702 702
703 703
704 704
705 705
706 706
707 707
708 708
709 709
710 710
711 711
712 712
713 713
714 714
715 715
716 716
717 717
718 718
719 719
720 720
721 721
722 722
723 723
724 724
725 725
726 726
727 727
728 728
729 729
730 730
731 731
732 732
733 733
734 734
735 735
736 736
737 737
738 738
739 739
740 740
741 741
742 742
743 743
744 744
745 745
746 746
747 747
748 748
749 749
750 750
751 751
752 752
753 753
754 754
755 755
756 756
757 757
758 758
759 759
760 760
761 761
762 762
763 763
764 764
765 765
766 766
767 767
768 768
769 769
770 770
771 771
772 772
773 773
774 774
775 775
776 776
777 777
778 778
779 779
780 780
781 781
782 782
783 783
784 784
785 785
786 786
787 787
788 788
789 789
790 790
791 791
792 792
793 793
794 794
795 795
796 796
797 797
798 798
799 799
800 800
801 801
802 802
803 803
804 804
805 805
806 806
807 807
808 808
809 809
810 810
811 811
812 812
813 813
814 814
815 815
816 816
817 817
818 818
819 819
820 820
821 821
822 822
823 823
824 824
825 825
826 826
827 827
828 828
829 829
830 830
831 831
832 832
833 833
834 834
835 835
836 836
837 837
838 838
839 839
840 840
841 841
842 842
843 843
844 844
845 845
846 846
847 847
848 848
849 849
850 850
851 851
852 852
853 853
854 854
855 855
856 856
857 857
858 858
859 859
860 860
861 861
862 862
863 863
864 864
865 865
866 866
867 867
868 868
869 869
870 870
871 871
872 872
873 873
874 874
875 875
876 876
877 877
878 878
879 879
880 880
881 881
882 882
883 883
884 884
885 885
886 886
887 887
888 888
889 889
890 890
891 891
892 892
893 893
894 894
895 895
896 896
897 897
898 898
899 899
900 900
901 901
902 902
903 903
904 904
905 905
906 906
907 907
908 908
909 909
910 910
911 911
912 912
913 913
914 914
915 915
916 916
917 917
918 918
919 919
920 920
921 921
922 922
923 923
924 924
925 925
926 926
927 927
928 928
929 929
930 930
931 931
932 932
933 933
934 934
935 935
936 936
937 937
938 938
939 939
940 940
941 941
942 942
943 943
944 944
945 945
946 946
947 947
948 948
949 949
950 950
951 951
952 952
953 953
954 954
955 955
956 956
957 957
958 958
959 959
960 960
961 961
962 962
963 963
964 964
965 965
966 966
967 967
968 968
969 969
970 970
971 971
972 972
973 973
974 974
975 975
976 976
977 977
978 978
979 979
980 980
981 981
982 982
983 983
984 984
985 985
986 986
987 987
988 988
989 989
990 990
991 991
992 992
993 993
994 994
995 995
996 996
997 997
998 998
999 999
1000 1000
1001 1001
1002 1002
1003 1003
1004 1004
1005 1005
1006 1006
1007 1007
1008 1008
1009 1009
1010 1010
1011 1011
1012 1012
1013 1013
1014 1014
1015 1015
1016 1016
1017 1017
1018 1018
1019 1019
1020 1020
1021 1021
1022 1022
1023 1023
1024 1024
1025 1025
1026 1026
1027 1027
1028 1028
1029 1029
1030 1030
1031 1031
1032 1032
1033 1033
1034 1034
1035 1035
1036 1036
1037 1037
1038 1038
1039 1039
1040 1040
1041 1041
1042 1042
1043 1043
1044 1044
1045 1045
1046 1046
1047 1047
1048 1048
1049 1049
1050 1050
1051 1051
1052 1052
1053 1053
1054 1054
1055 1055
1056 1056
1057 1057
1058 1058
1059 1059
1060 1060
1061 1061
1062 1062
1063 1063
1064 1064
1065 1065
1066 1066
1067 1067
1068 1068
1069 1069
1070 1070
1071 1071
1072 1072
1073 1073
1074 1074
1075 1075
1076 1076
1077 1077
1078 1078
1079 1079
1080 1080
1081 1081
1082 1082
1083 1083
1084 1084
1085 1085
1086 1086
1087 1087
1088 1088
1089 1089
1090 1090
1091 1091
1092 1092
1093 1093
1094 1094
1095 1095
1096 1096
1097 1097
1098 1098
1099 1099
1100 1100
1101 1101
1102 1102
1103 1103
1104 1104
1105 1105
1106 1106
1107 1107
1108 1108
1109 1109
1110 1110
1111 1111
1112 1112
1113 1113
1114 1114
1115 1115
1116 1116
1117 1117
1118 1118
1119 1119
1120 1120
1121 1121
1122 1122
1123 1123
1124 1124
1125 1125
1126 1126
1127 1127
1128 1128
1129 1129
1130 1130
1131 1131
1132 1132
1133 1133
1134 1134
1135 1135
1136 1136
1137 1137
1138 1138
1139 1139
1140 1140
1141 1141
1142 1142
1143 1143
1144 1144
1145 1145
1146 1146
1147 1147
1148 1148
1149 1149
1150 1150
1151 1151
1152 1152
1153 1153
1154 1154
1155 1155
1156 1156
1157 1157
1158 1158
1159 1159
1160 1160
1161 1161
1162 1162
1163 1163
1164 1164
1165 1165
1166 1166
1167 1167
1168 1168
1169 1169
1170 1170
1171 1171
1172 1172
1173 1173
1174 1174
1175 1175
1176 1176
1177 1177
1178 1178
1179 1179
1180 1180
1181 1181
1182 1182
1183 1183
1184 1184
1185 1185
1186 1186
1187 1187
1188 1188
1189 1189
1190 1190
1191 1191
1192 1192
1193 1193
1194 1194
1195 1195
1196 1196
1197 1197
1198 1198
1199 1199
1200 1200
1201 1201
1202 1202
1203 1203
1204 1204
1205 1205
1206 1206
1207 1207
1208 1208
1209 1209
1210 1210
1211 1211
1212 1212
1213 1213
1214 1214
1215 1215
1216 1216
1217 1217
1218 1218
1219 1219
1220 1220
1221 1221
1222 1222
1223 1223
1224 1224
1225 1225
1226 1226
1227 1227
1228 1228
1229 1229
1230 1230
1231 1231
1232 1232
1233 1233
1234 1234
1235 1235
1236 1236
1237 1237
1238 1238
1239 1239
1240 1240
1241 1241
1242 1242
1243 1243
1244 1244
1245 1245
1246 1246
1247 1247
1248 1248
1249 1249
1250 1250
1251 1251
1252 1252
1253 1253
1254 1254
1255 1255
1256 1256
1257 1257
1258 1258
1259 1259
1260 1260
1261 1261
1262 1262
1263 1263
1264 1264
1265 1265
1266 1266
1267 1267
1268 1268
1269 1269
1270 1270
1271 1271
1272 1272
1273 1273
1274 1274
1275 1275
1276 1276
1277 1277
1278 1278
1279 1279
1280 1280
1281 1281
1282 1282
1283 1283
1284 1284
1285 1285
1286 1286
1287 1287
1288 1288
1289 1289
1290 1290
1291 1291
1292 1292
1293 1293
1294 1294
1295 1295
1296 1296
1297 1297
1298 1298
1299 1299
1300 1300
1301 1301
1302 1302
1303 1303
1304 1304
1305 1305
1306 1306
1307 1307
1308 1308
1309 1309
1310 1310
1311 1311
1312 1312
1313 1313
1314 1314
1315 1315
1316 1316
1317 1317
1318 1318
1319 1319
1320 1320
1321 1321
1322 1322
1323 1323
1324 1324
1325 1325
1326 1326
1327 1327
1328 1328
1329 1329
1330 1330
1331 1331
1332 1332
1333 1333
1334 1334
1335 1335
1336 1336
1337 1337
1338 1338
1339 1339
1340 1340
1341 1341
1342 1342
1343 1343
1344 1344
1345 1345
1346 1346
1347 1347
1348 1348
1349 1349
1350 1350
1351 1351
1352 1352
1353 1353
1354 1354
1355 1355
1356 1356
1357 1357
1358 1358
1359 1359
1360 1360
1361 1361
1362 1362
1363 1363
1364 1364
1365 1365
1366 1366
1367 1367
1368 1368
1369 1369
1370 1370
1371 1371
1372 1372
1373 1373
1374 1374
1375 1375
1376 1376
1377 1377
1378 1378
1379 1379
1380 1380
1381 1381
1382 1382
1383 1383
1384 1384
1385 1385
1386 1386
1387 1387
1388 1388
1389 1389
1390 1390
1391 1391
1392 1392
1393 1393
1394 1394
1395 1395
1396 1396
1397 1397
1398 1398
1399 1399
1400 1400
1401 1401
1402 1402
1403 1403
1404 1404
1405 1405
1406 1406
1407 1407
1408 1408
1409 1409
1410 1410
1411 1411
1412 1412
1413 1413
1414 1414
1415 1415
1416 1416
14
```

# Google Assistant Developer Console:-

The screenshot shows the Google Assistant Developer Console in the 'Develop' tab. A custom intent named 'hall\_light\_off' is being edited. The interface includes:

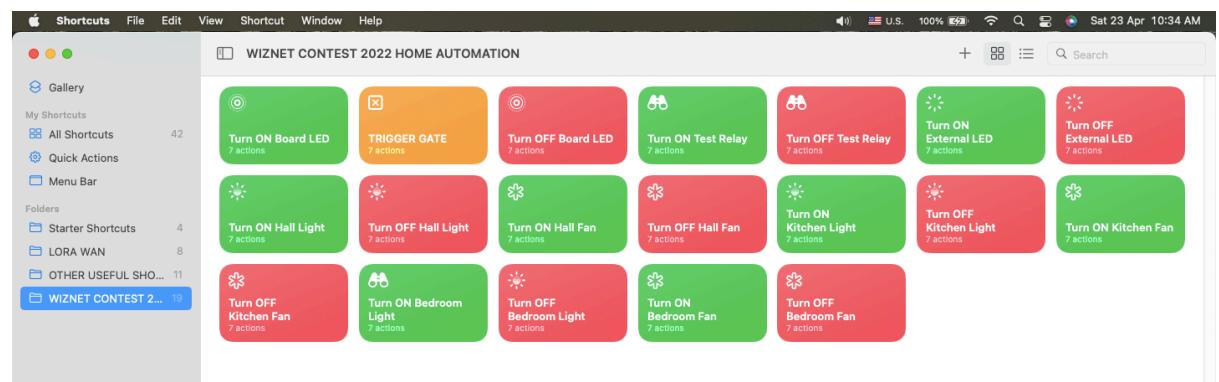
- Is this a global intent?**: A question with 'Yes' and 'No' radio buttons, and a 'Edit global intent handling' button.
- Add training phrases**: A section to enter at least 10 phrases a user might say to match this intent.
- Switch OFF Hall Light**: A sample phrase 'Turn OFF Hall Light'.
- Add intent parameters**: A table to define parameters for the intent.
- Available intents**: A sidebar listing various intents like 'bedroom\_fan\_off', 'bedroom\_fan\_on', etc.



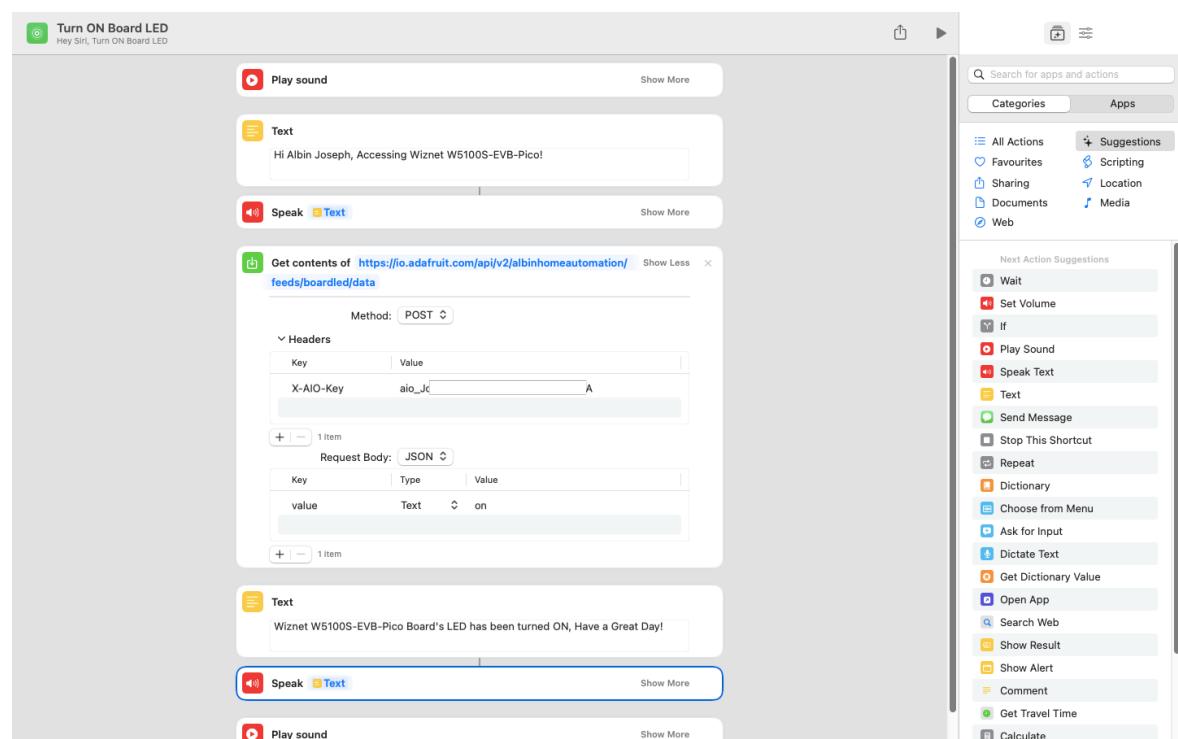
## 8) Sample Shortcuts Code and Shortcuts Achievement

The below pictures shows the shortcuts dashboard and a few code snippets:-

### Siri Shortcuts Dashboard:-



### Sample Shortcuts Code POST Method:-



The most difficult part faced in Apple shortcuts was to make a proper POST request which would send the value to Adafruit IO, usually people just make use of a 3<sup>rd</sup> party service called IFTTT to get a webhook which actually behind the scenes perform a post request however the correct API format to make a https based POST request via Apple shortcuts is unavailable on the internet which seemed to be a small but great achievement to completely get rid of any 3<sup>rd</sup> party service especially IFTTT.

## 9) Circuit Python Code and Adafruit IO Dashboard

The Board was programmed using Circuit python, there were a few difficulties where the code was unresponsive and was required to be nuked with the flash nuke.uf2 file.

The editor that was widely used for this project was the Mu Editor and sometimes Visual Studio Code seemed easier for search. Below is the code used in the project:-

```

43. SPI0_CSn = board.GP17
44.
45. #Reset
46. W5x00_RSTn = board.GP20
47.
48. print(" ""HOME AUTOMATION" "\n"      " WIZNET CONTEST" "\n"      " " "2022")
49. sleep(4)
50. print("\n" "    Created by" "\n"      " " "Albin Joseph")
51. sleep(4)
52.
53. print("Pinging Adafruit           IO""\n"" Wiznet5k (DHCP)")
54. # Setup your network configuration below
55. # random MAC, later should change this value on your vendor ID
56. MY_MAC = (0x00, 0x01, 0x02, 0x03, 0x04, 0x05)
57. IP_ADDRESS = (192, 168, 1, 100)
58. SUBNET_MASK = (255, 255, 255, 0)
59. GATEWAY_ADDRESS = (192, 168, 1, 1)
60. DNS_SERVER = (8, 8, 8, 8)
61.
62. ethernetRst = digitalio.DigitalInOut(W5x00_RSTn)
63. ethernetRst.direction = digitalio.Direction.OUTPUT
64.
65. # For Adafruit Ethernet FeatherWing
66. cs = digitalio.DigitalInOut(SPI0_CSn)
67. # For Particle Ethernet FeatherWing
68. # cs = digitalio.DigitalInOut(board.D5)
69.
70. spi_bus = busio.SPI(SPI0_SCK, MOSI=SPI0_TX, MISO=SPI0_RX)
71.
72. # Reset W5x00 first
73. ethernetRst.value = False
74. time.sleep(1)
75. ethernetRst.value = True
76.
77. # Initialize ethernet interface with DHCP
78. eth = WIZNET5K(spi_bus, cs, is_dhcp=True, mac=MY_MAC, debug=False)
79.
80. print("Chip Version:", eth.chip)
81. print("MAC Address:", [hex(i) for i in eth.mac_address])
82. print("My IP address is:", eth.pretty_ip(eth.ip_address))
83.
84. ### Code ###
85. # Define callback methods which are called when events occur
86. # pylint: disable=unused-argument, redefined-outer-name
87. def connected(clinet):
88.     # This function will be called when the mqtt_client is connected
89.     # successfully to the broker.
90.     print("Connected to Adafruit IO!")
91.
92.     # Subscribe to Group
93.     io.subscribe(group_key=group_name)
94.
95. def disconnected(clinet):
96.     # This method is called when the mqtt_client disconnects
97.     # from the broker.
98.     print("Disconnected from Adafruit IO!")
99.
100. def subscribe(client, userdata, topic, granted_qos):
101.     # This method is called when the client subscribes to a new feed.
102.     print("Subscribed to {0} with QOS level {1}.format(topic, granted_qos))")
103.
104. def message(client, topic, message):
105.     # Method called when a client's subscribed feed has a new value.
106.     print(" {0}: {1}.format(topic, message))")
107.
108. # Board LED CONTROL
109. ///////////////////////////////////////////////////////////////////
110. BoardLed = digitalio.DigitalInOut(board.GP25)
111. BoardLed.direction = digitalio.Direction.OUTPUT

```

```

112. def on_BoardLed_msg(client, topic, message):
113.     # Method called when a client's subscribed feed has a new value.
114.     print(" {0}: {1}".format(topic, message))
115.     if message == "on":
116.         BoardLed.value = True
117.     elif message == "off":
118.         BoardLed.value = False
119.     else:
120.         print("Unexpected message on BoardLed feed")
121.
122. # External LED CONTROL
123. ///////////////////////////////////////////////////
124. ExternalLed = digitalio.DigitalInOut(board.GP2)
125. ExternalLed.direction = digitalio.Direction.OUTPUT
126.
127. def on_ExternalLed_msg(client, topic, message):
128.     # Method called when a client's subscribed feed has a new value.
129.     print(" {0}: {1}".format(topic, message))
130.     if message == "on":
131.         ExternalLed.value = True
132.     elif message == "off":
133.         ExternalLed.value = False
134.     else:
135.         print("Unexpected message on ExternalLed feed")
136. # KITCHEN FAN CONTROL
137. ///////////////////////////////////////////////////
138. KitchenFan = digitalio.DigitalInOut(board.GP7)
139. KitchenFan.direction = digitalio.Direction.OUTPUT
140.
141. def on_KitchenFan_msg(client, topic, message):
142.     # Method called when a client's subscribed feed has a new value.
143.     print(" {0}: {1}".format(topic, message))
144.     if message == "on":
145.         KitchenFan.value = True
146.     elif message == "off":
147.         KitchenFan.value = False
148.     else:
149.         print("Unexpected message on KitchenFan feed")
150. # BEDROOM FAN CONTROL
151. ///////////////////////////////////////////////////
152. BedroomFan = digitalio.DigitalInOut(board.GP6)
153. BedroomFan.direction = digitalio.Direction.OUTPUT
154.
155. def on_BedroomFan_msg(client, topic, message):
156.     # Method called when a client's subscribed feed has a new value.
157.     print(" {0}: {1}".format(topic, message))
158.     if message == "on":
159.         BedroomFan.value = True
160.     elif message == "off":
161.         BedroomFan.value = False
162.     else:
163.         print("Unexpected message on BedroomFan feed")
164. # HALL FAN CONTROL GPIO PIN CONTROL
165. ///////////////////////////////////////////////////
166. # HallFan = digitalio.DigitalInOut(board.GP22)
167. # HallFan.direction = digitalio.Direction.OUTPUT
168.
169. # def on_HallFan_msg(client, topic, message):
170. #     # Method called when a client's subscribed feed has a new value.
171. #     print(" {0}: {1}".format(topic, message))
172. #     if message == "on":
173. #         HallFan.value = True
174. #     elif message == "off":
175. #         HallFan.value = False
176. #     else:
177. #         print("Unexpected message on HallFan feed")

```

```

178.
179. #
180. # //////////////// Relay Test /////////////////////////////////
181. # Relay CONTROL System (Low/False is ON
182. # //////////////// RelayTest \\\\\\\\
183.
184.
185. RelayTest = digitalio.DigitalInOut(board.GP9)
186. RelayTest.direction = digitalio.Direction.OUTPUT
187. RelayTest.value = True
188.
189. def on_RelayTest_msg(client, topic, message):
190.     # Method called when a client's subscribed feed has a new value.
191.     print(" {0}: {1}".format(topic, message))
192.     if message == "on":
193.         RelayTest.value = False
194.     elif message == "off":
195.         RelayTest.value = True
196.     else:
197.         print("Unexpected message on RelayTest feed")
198.
199. # //////////////// HALL FAN CONTROL (RELAY) \\\\\\\\
200.
201. HallFan = digitalio.DigitalInOut(board.GP15)
202. HallFan.direction = digitalio.Direction.OUTPUT
203. HallFan.value = True
204.
205. def on_HallFan_msg(client, topic, message):
206.     ## Method called when a client's subscribed feed has a new value.
207.     print(" {0}: {1}".format(topic, message))
208.     if message == "on":
209.         HallFan.value = False
210.     elif message == "off":
211.         HallFan.value = True
212.     else:
213.         print("Unexpected message on HallFan feed")
214.
215. # HallFanRelay = digitalio.DigitalInOut(board.GP15)
216. # HallFanRelay.direction = digitalio.Direction.OUTPUT
217. # HallFanRelay.value = True
218.
219. # def on_HallFanRelay_msg(client, topic, message):
220.     ## Method called when a client's subscribed feed has a new value.
221.     #     print(" {0}: {1}".format(topic, message))
222.     #     if message == "on":
223.     #         HallFanRelay.value = False
224.     #     elif message == "off":
225.     #         HallFanRelay.value = True
226.     #     else:
227.     #         print("Unexpected message on HallFanRelay feed")
228.
229. # //////////////// HALL LIGHT \\\\\\\\
230.
231. HallLight = digitalio.DigitalInOut(board.GP14)
232. HallLight.direction = digitalio.Direction.OUTPUT
233. HallLight.value = True
234.
235. def on_HallLight_msg(client, topic, message):
236.     # Method called when a client's subscribed feed has a new value.
237.     print(" {0}: {1}".format(topic, message))
238.     if message == "on":
239.         HallLight.value = False
240.     elif message == "off":
241.         HallLight.value = True
242.     else:
243.         print("Unexpected message on HallLight feed")
244.

```

```

245. # ///////// BEDROOM LIGHT \\\\\\\\
246.
247. BedroomLight = digitalio.DigitalInOut(board.GP13)
248. BedroomLight.direction = digitalio.Direction.OUTPUT
249. BedroomLight.value = True
250.
251. def on_BedroomLight_msg(client, topic, message):
252.     # Method called when a client's subscribed feed has a new value.
253.     print(" {0}: {1}".format(topic, message))
254.     if message == "on":
255.         BedroomLight.value = False
256.     elif message == "off":
257.         BedroomLight.value = True
258.     else:
259.         print("Unexpected message on BedroomLight feed")
260.
261. # ///////// KITCHEN LIGHT \\\\\\\\
262.
263. KitchenLight = digitalio.DigitalInOut(board.GP12)
264. KitchenLight.direction = digitalio.Direction.OUTPUT
265. KitchenLight.value = True
266.
267. def on_KitchenLight_msg(client, topic, message):
268.     # Method called when a client's subscribed feed has a new value.
269.     print(" {0}: {1}".format(topic, message))
270.     if message == "on":
271.         KitchenLight.value = False
272.     elif message == "off":
273.         KitchenLight.value = True
274.     else:
275.         print("Unexpected message on KitchenLight feed")
276.
277. # *****GATE AUTOMATION*****
278. # //////// GateButton \\\\\\\\
279.
280. GateButton = digitalio.DigitalInOut(board.GP8)
281. GateButton.direction = digitalio.Direction.OUTPUT
282. GateButton.value = True
283.
284. def on_GateButton_msg(client, topic, message):
285.     # Method called when a client's subscribed feed has a new value.
286.     print(" {0}: {1}".format(topic, message))
287.     if message == "on":
288.         GateButton.value = False
289.         sleep(1)
290.         GateButton.value = True
291.         print("Gate triggered!!")
292.         message == "off"
293.     else:
294.         print("GATE Triggered" "\n" "with Adafruit IO" "\n" "    !!!")
295.
296. # /////////////////////////////////
297.
298. # Initialize MQTT interface with the ethernet interface
299. MQTT.set_socket(socket, eth)
300.
301. # Initialize a new MQTT Client object
302. mqtt_client = MQTT.MQTT(
303.     broker="io.adafruit.com",
304.     username=secrets["aio_username"],
305.     password=secrets["aio_key"],
306.     is_ssl=False,
307. )
308.
309. # Initialize an Adafruit IO MQTT Client
310. io = IO_MQTT(mqtt_client)
311.
312. # Setup the callback methods above
313. io.on_connect = connected
314. io.on_disconnect = disconnected

```

```

315. io.on_message = message
316. io.on_subscribe = subscribe
317.
318. # Set up a callback for the BoardLed feed /////////////////////////////////
319. io.add_feed_callback("BoardLed", on_BoardLed_msg)
320.
321. # Set up a callback for the ExternalLed feed
322. ///////////////////////////////////////////////////
323. io.add_feed_callback("ExternalLed", on_ExternalLed_msg)
324.
325. # Set up a callback for the KitchenFan feed /////////////////////////////////
326. io.add_feed_callback("KitchenFan", on_KitchenFan_msg)
327.
328. # Set up a callback for the BedroomFan feed /////////////////////////////////
329. io.add_feed_callback("BedroomFan", on_BedroomFan_msg)
330.
331. # Set up a callback for the HallFan feed /////////////////////////////////
332. io.add_feed_callback("HallFan", on_HallFan_msg)
333.
334. # /////////////////////////////////////////////////
335. # Set up a callback for the RelayTest feed /////////////////////////////////
336. io.add_feed_callback("RelayTest", on_RelayTest_msg)
337.
338. ## Set up a callback for the HallFanRelay feed
339. ///////////////////////////////////////////////////
340. # io.add_feed_callback("HallFanRelay", on_HallFanRelay_msg)
341.
342. # Set up a callback for the HallLight feed /////////////////////////////////
343. io.add_feed_callback("HallLight", on_HallLight_msg)
344.
345. # Set up a callback for the BedroomLight feed
346. ///////////////////////////////////////////////////
347. io.add_feed_callback("BedroomLight", on_BedroomLight_msg)
348.
349. # Set up a callback for the KitchenLight feed
350. ///////////////////////////////////////////////////
351. io.add_feed_callback("KitchenLight", on_KitchenLight_msg)
352.
353. ## Group name
354. group_name = "weatherstation" # Comenting Throws error need to check
355.
356. ## Feeds within the group
357. temp_feed = "weatherstation.temperature" # omenting Throws error need to check
358. humid_feed = "weatherstation.humidity" # Comenting Throws error need to check
359.
360. # Connect to Adafruit IO
361. print("Connecting to Adafruit IO...")
362. io.connect()
363.
364. # ////////////////////////////////ADD ON
365. SUBSCRIPTION///////////////////////////////
366. # # Subscribe to all messages on the BoardLed feed
367. io.subscribe("BoardLed")
368.
369. # # Subscribe to all messages on the ExternalLed feed
370. io.subscribe("ExternalLed")
371.
372. # Subscribe to all messages on the KitchenFan feed
373. io.subscribe("KitchenFan")
374.
375. # Subscribe to all messages on the BedroomFan feed
376. io.subscribe("BedroomFan")
377.
378. # # Subscribe to all messages on the HallFan feed
379. io.subscribe("HallFan")

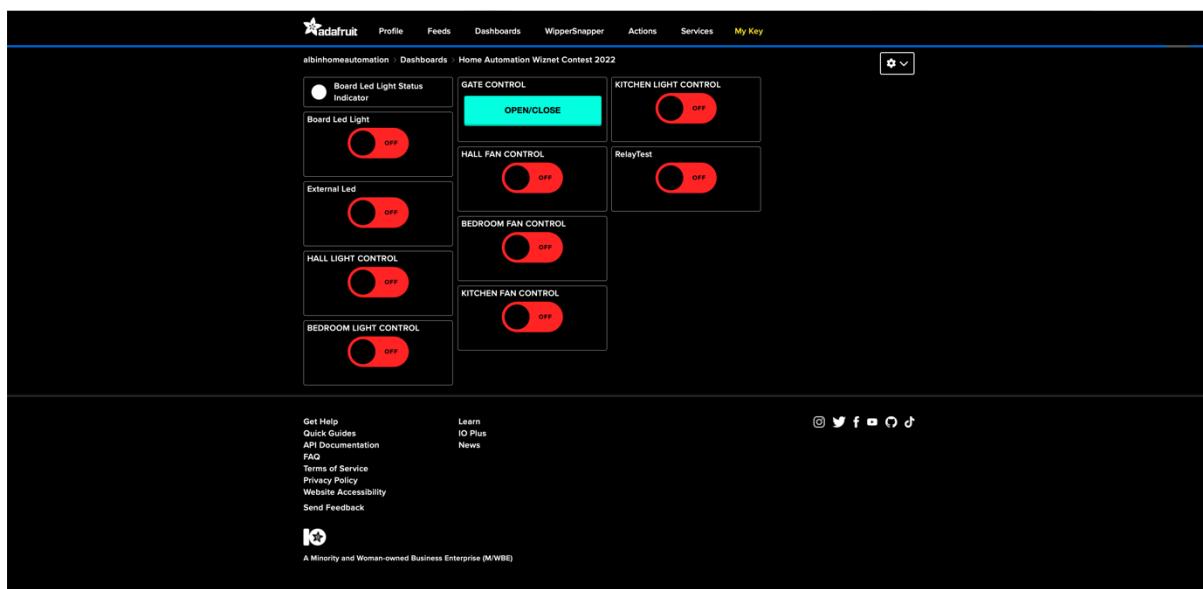
```

```

380.
381. # ////////////////////////////////RELAY
      SUBSCRIPTION///////////////////////////////
382.
383. # # Subscribe to all messages on the RelayTest feed
384. io.subscribe("RelayTest")
385.
386. # Subscribe to all messages on the HallFanRelay feed
387. # io.subscribe("HallFanRelay")
388.
389. # # Subscribe to all messages on the HallLight feed
390. io.subscribe("HallLight")
391.
392. # # Subscribe to all messages on the BedroomLight feed
393. io.subscribe("BedroomLight")
394.
395. # # Subscribe to all messages on the KitchenLight feed
396. io.subscribe("KitchenLight")
397.
398. # # Subscribe to all messages on the GateButton feed
399. io.subscribe("GateButton")
400.
401. # /////////////////////////////////
402.
403. print("Connected to Adafruit IO")
404. print("    HandShake      SUCCESS! !")
405. print("WAITING FOR INPUT")
406.
407. while True:
408.     io.loop()
409.

```

## Adafruit IO Dashboard:-



## **10) Results and Conclusion**

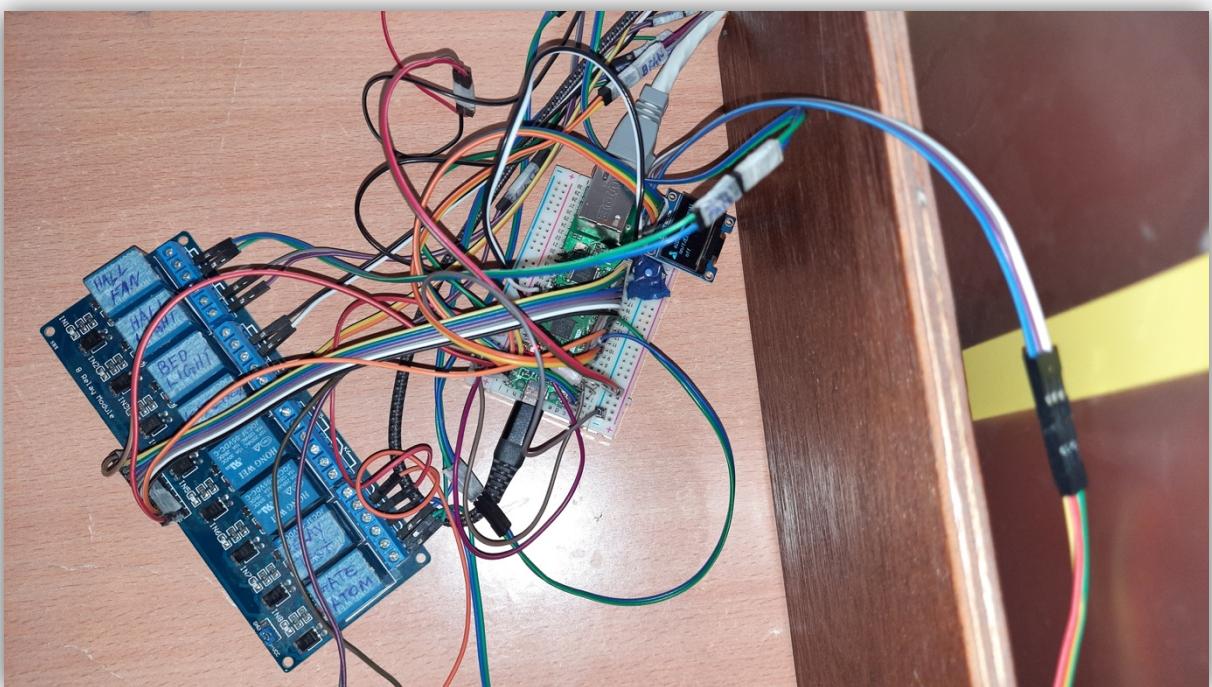
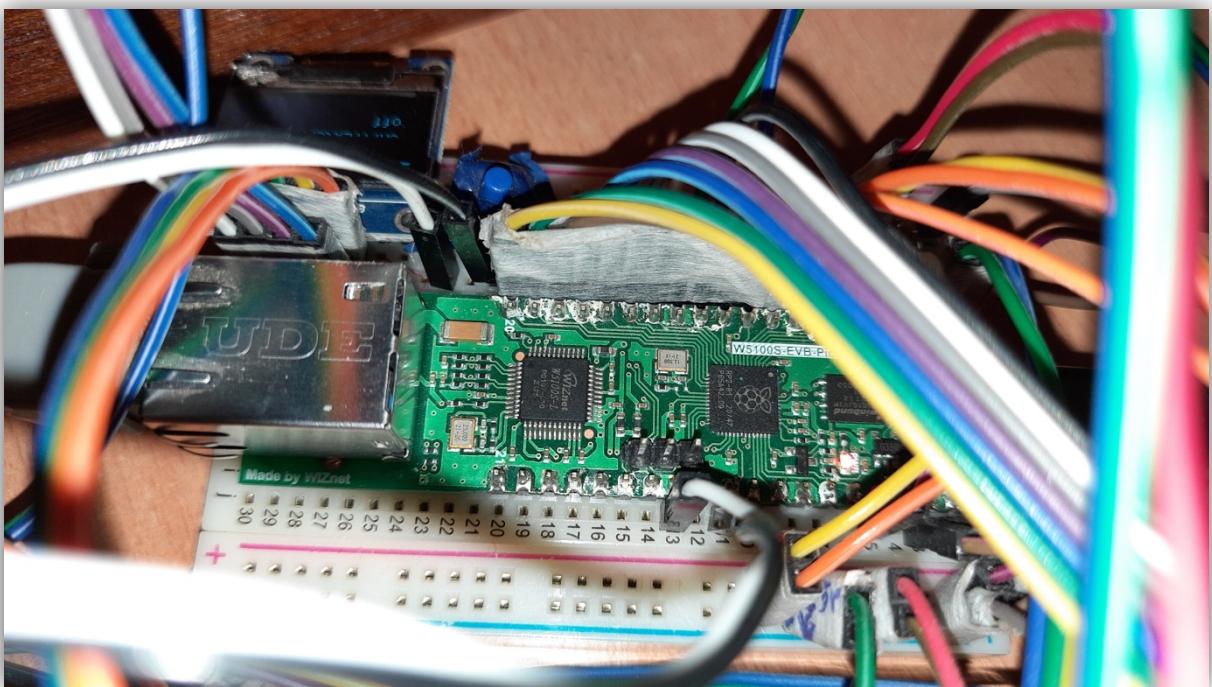
Well the results were great! Automated gates lights and fans all powered by the lone board W5500-EVB-Pico programmed with circuit python, connected to Adafruit IO which was a possibility only because of Wiznet's brilliant idea of adding an Ethernet (RJ45) port powered by the great W5500 hardwired TCP/IP chip and equally important wiznet 5k ethernet library and adafruit library which inspired and helped me complete this amazing project successfully.

I was also able to learn and gain a lot of experience from this project and I'm really grateful for this grand opportunity wiznet provided to us.

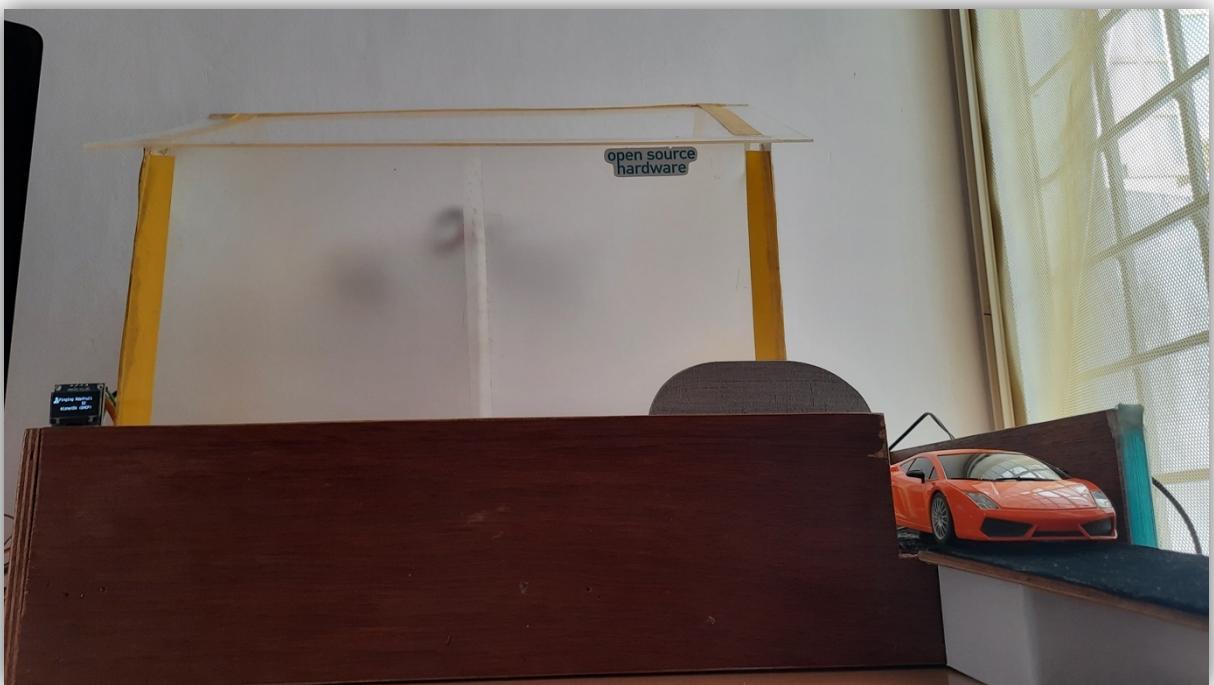
### **Please Note:-**

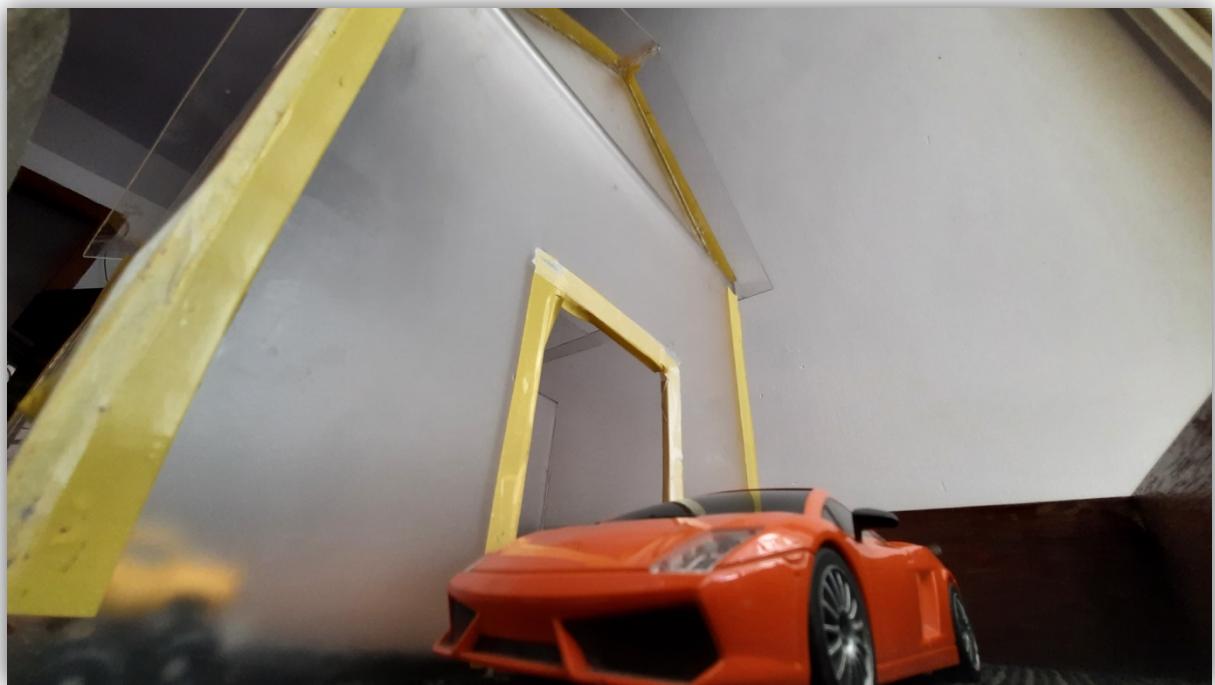
A video on this project including a working demo is being prepared which will be shared soon.

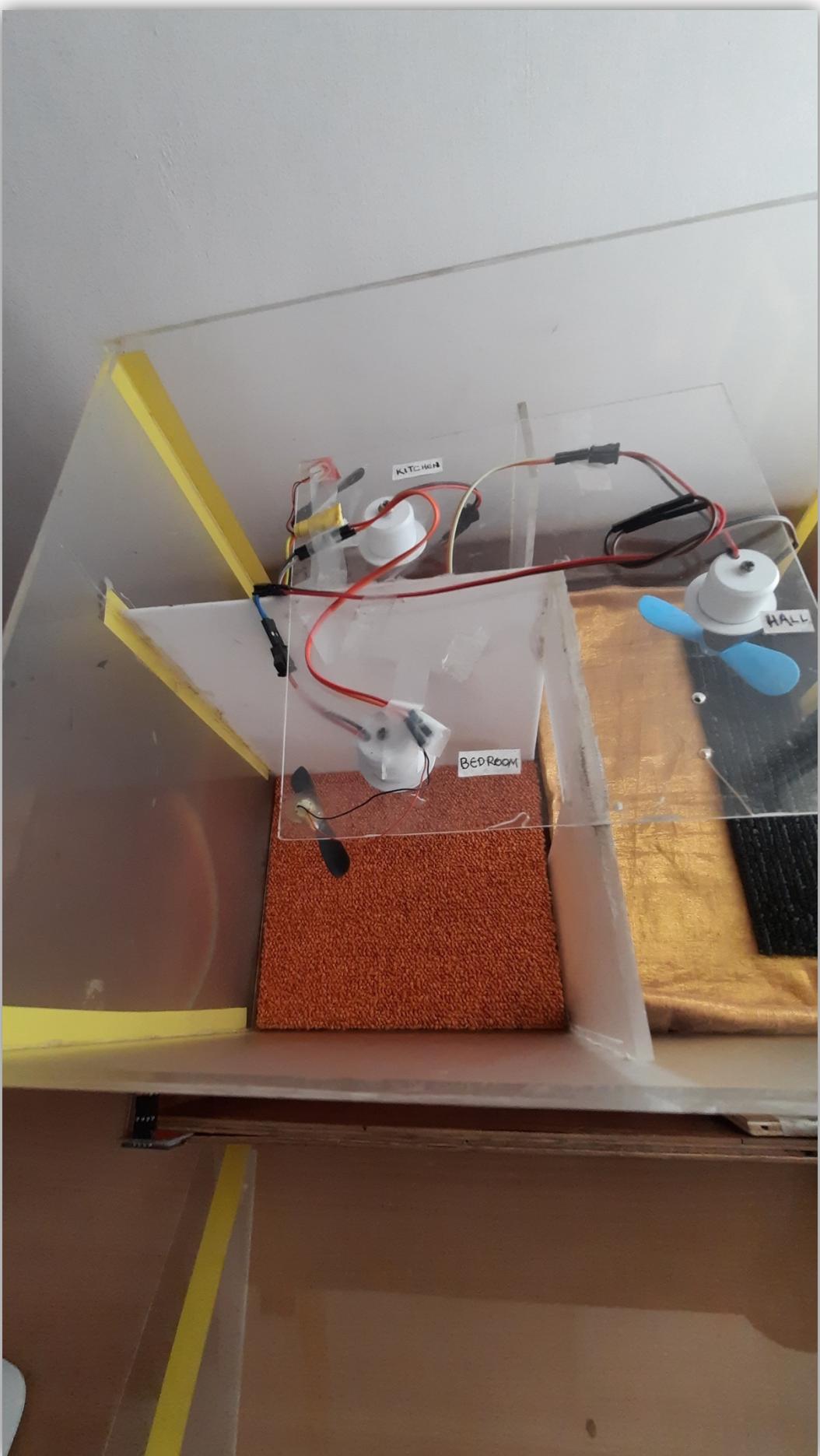
The below images in the following pages shows some images that I have captured :-

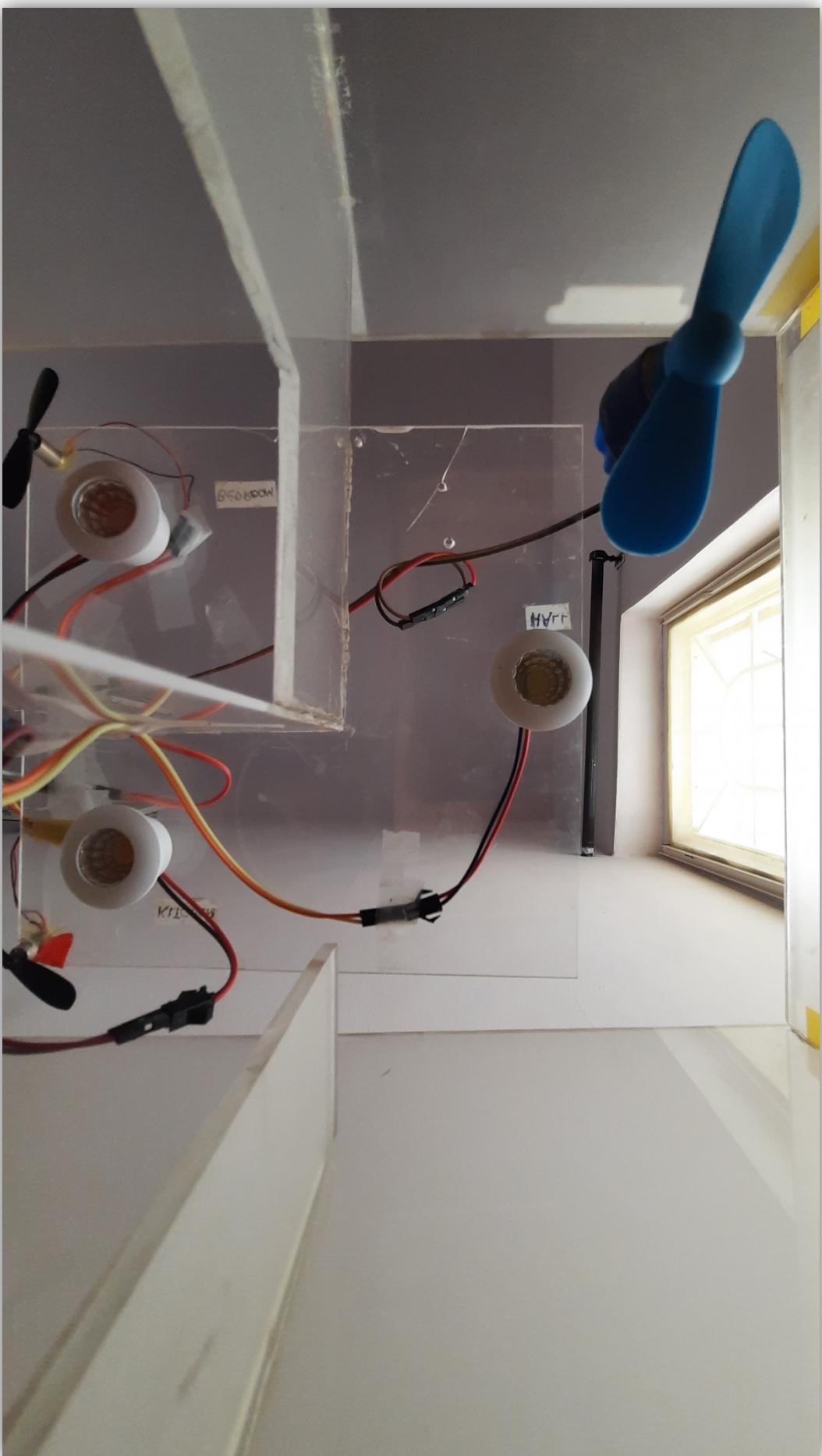




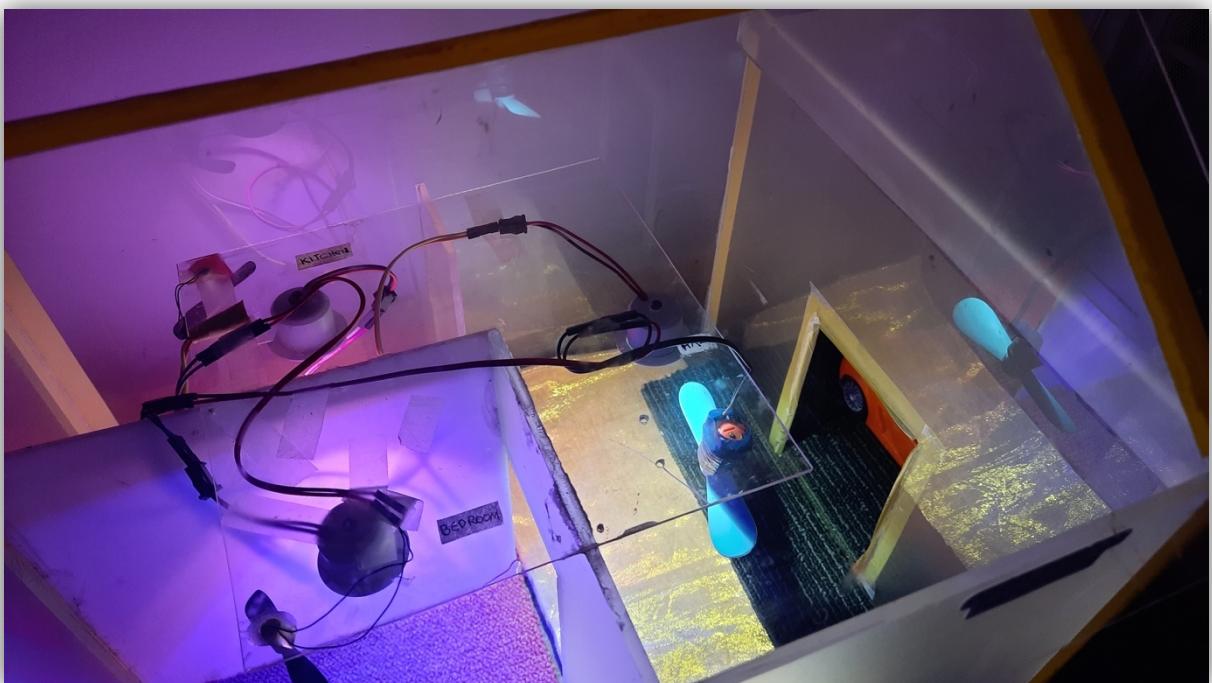
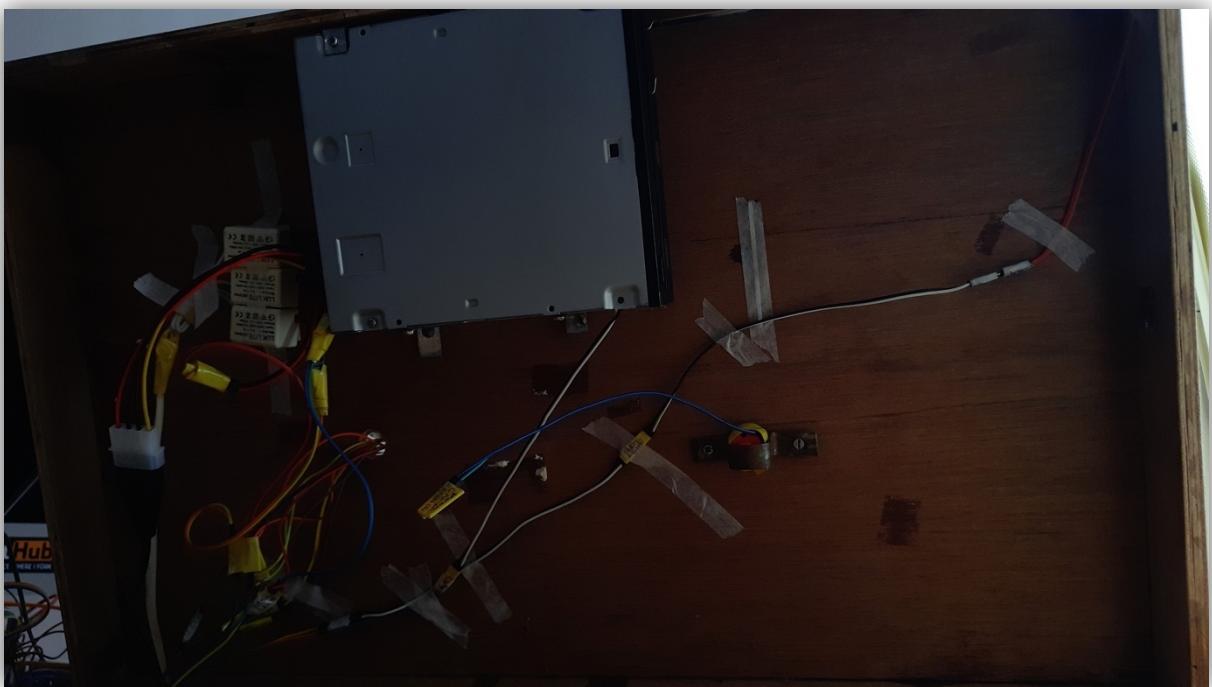








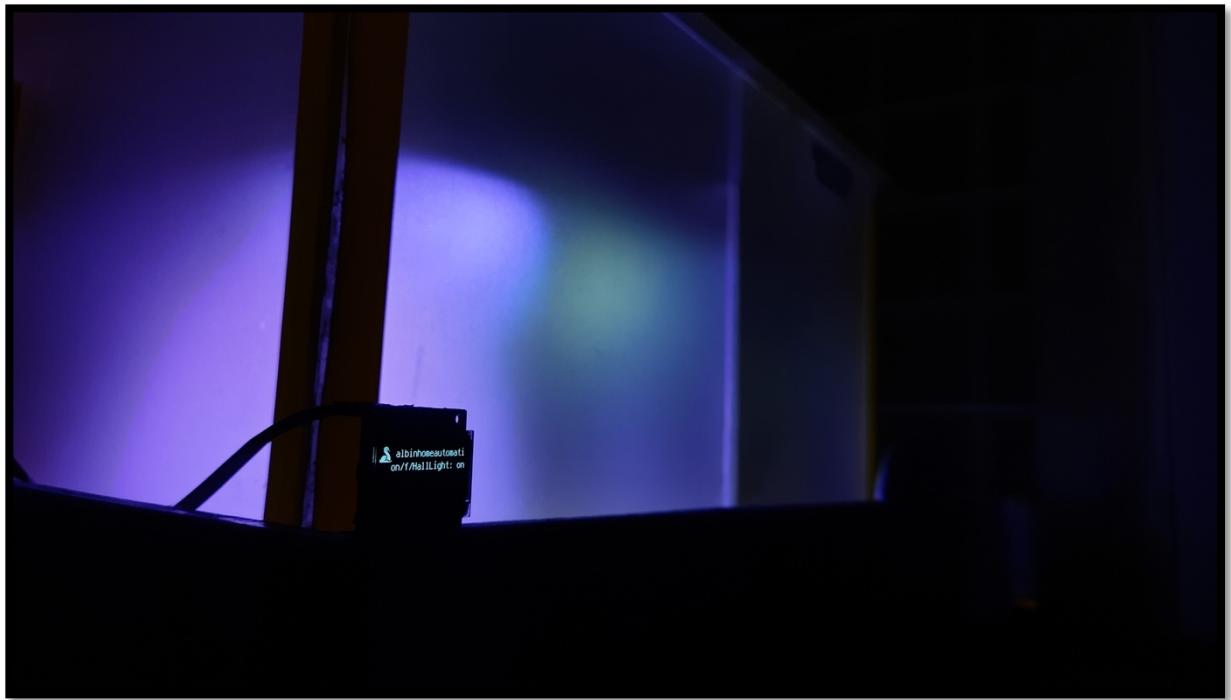












# **THANK YOU**

**END OF DOCUMENT**