



## Program Coding Pada Python.

```

51 cards = [
52     {
53         "id": "40489236889",
54         "rack": None,
55         "items": 0
56     },
57     {
58         "id": "40031248656",
59         "rack": None,
60         "items": 0
61     },
62     {
63         "id": "100074070001",
64         "rack": None,
65         "items": 0
66     },
67     {
68         "id": "80311825888",
69         "rack": None,
70         "items": 0
71     },
72     {
73         "id": "40901329783",
74         "rack": None,
75         "items": 0
76     },
77     {
78         "id": "31530958549",
79         "rack": None,
80         "items": 0
81     },
82     {
83         "id": "31530958549",
84         "rack": None,
85         "items": 0
86     }
87 ]
88
89 #initial setup
90 def setup_buzzer():
91     GPIO.setup(buzzer, GPIO.OUT)
92     GPIO.output(buzzer, GPIO.LOW)
93     time.sleep(1)
94
95 #setup buzzer
96 def beep(n=1):
97     for i in range(n):
98         GPIO.output(buzzer, GPIO.HIGH)
99         time.sleep(0.1)
100        GPIO.output(buzzer, GPIO.LOW)
101        time.sleep(0.1)

```

20:37:29: This is Geany 1.37.1.

line: 59 / 481 col: 21 sel: 0 INS SP mode: LF encoding: UTF-8 filetype: Python scope: unknown

```

101 #clean up GPIO
102 def tearDown():
103     GPIO.cleanup()
104
105 #initialized motors
106 def init_motors():
107     global motors
108     global switch_X
109     global switch_Y
110     global switch_Z
111
112     # motor X
113     GPIO.setup(switch_X, GPIO.IN, pull_up_down=GPIO.PUD_UP)
114     GPIO.add_event_detect(switch_X, GPIO.FALLING, callback=stop_motor)
115
116     # motor Y
117     GPIO.setup(switch_Y, GPIO.IN, pull_up_down=GPIO.PUD_UP)
118     GPIO.add_event_detect(switch_Y, GPIO.FALLING, callback=stop_motor)
119
120     # motor Z
121     GPIO.setup(switch_Z, GPIO.IN, pull_up_down=GPIO.PUD_UP)
122     GPIO.add_event_detect(switch_Z, GPIO.RISING, callback=stop_motor)
123
124     for k, v in motors.items():
125         if k != "bz":
126             v["cls"] = RpiMotorLib.A4988Nema(v["direction"], v["step"], v["gpio"], "A4988")
127         else:
128             for i in v:
129                 v["cls"] = RpiMotorLib.A4988Nema(i["direction"], i["step"], i["gpio"], "A4988")
130
131     setup_buzzer()
132     print("Set starting point...")
133     set_starting_point()
134     time.sleep(3)
135
136 # stop motor
137 def stop_motor(c):
138     global motors
139     global motor_X_state
140     global motor_Y_state
141     global motor_Z_state
142
143     inp = GPIO.input(c)
144     state = None
145
146

```

20:37:29: This is Geany 1.37.1.

line: 59 / 481 col: 21 sel: 0 INS SP mode: LF encoding: UTF-8 filetype: Python scope: unknown

```

smartinventory.py - /home/raspberry/project/smartinventory - Geany
File Edit Search View Document Project Build Tools Help
Documents > requirements.txt x rfid.py x shape.py x smartinventory.py x shape_finder.py x motordc.py x motor.py x multimotor.py x button.py x
-|project
  |inventory.py
  |project...tinventory
  |button.py
  |motor.py
  |motordc.py
  |multimotor.py
  |requirements.txt
  |rfid.py
  |shape.py
  |shape_finder.py
  |smartinventory.py
151 motor = None
152
153 if c == 12:
154     motor = motors["mx"]
155     state = motor_x_state
156 elif c == 18:
157     motor = motors["my"]
158     state = motor_y_state
159 else:
160     motor = motors["mz"]
161     state = motor_z_state
162
163 if motor and inp == 1 and state == 1:
164     if c == 12:
165         motors["mx"]["cls"].motor_stop()
166         motors["mx"]["cls"] = None
167         motor_x_state = 2
168     elif c == 18:
169         motors["my"]["cls"].motor_stop()
170         motors["my"]["cls"] = None
171         motor_y_state = 2
172     else:
173         motors["mz"][0]["cls"].motor_stop()
174         motors["mz"][1]["cls"].motor_stop()
175         motors["mz"][0]["cls"] = None
176         motors["mz"][1]["cls"] = None
177         motor_z_state = 2
178
179     time.sleep(2)
180
181
182 #set starting point
183 def set_starting_point():
184
185     global switch_X
186     global switch_Y
187     global switch_Z
188
189     #check motor z
190     INPUT_Z = GPIO.input(switch_Z)
191     if INPUT_Z == 0:
192         run_motor("mz", 0, 3700, True)
193
194     time.sleep(1)
195
196     #check motor y
197     INPUT_Y = GPIO.input(switch_Y)
198     if INPUT_Y == 0:
199         run_motor("my", 0, 8000)
200
201     time.sleep(1)
202
20:37:29: This is Geany 1.37.1.
line: 179 / 481 col: 21 sel: 0 INS SP mode: LF encoding: UTF-8 filetype: Python scope: stop_motor

```

```

smartinventory.py - /home/raspberry/project/smartinventory - Geany
File Edit Search View Document Project Build Tools Help
Documents > requirements.txt x rfid.py x shape.py x smartinventory.py x shape_finder.py x motordc.py x motor.py x multimotor.py x button.py x
-|project
  |inventory.py
  |project...tinventory
  |button.py
  |motor.py
  |motordc.py
  |multimotor.py
  |requirements.txt
  |rfid.py
  |shape.py
  |shape_finder.py
  |smartinventory.py
251 motor_motor_go(d, "Full", s, .0005, True, .05)
252     time.sleep(1)
253
254 else:
255     inp = GPIO.input(switch_Z)
256
257     if inp == 1:
258         motor_z_state = 2
259     else:
260         motor_z_state = 1
261
262     if set_point:
263         direction_1 = True
264         direction_2 = False
265     else:
266         direction_1 = False
267         direction_2 = True
268
269     m21 = motors["mz"][0]["cls"]
270     m22 = motors["mz"][1]["cls"]
271
272     with concurrent.futures.ThreadPoolExecutor() as executor:
273         f1 = executor.submit(m21.motor_go, direction_1, "Full", s, 0.0005, True, .05)
274         f2 = executor.submit(m22.motor_go, direction_2, "Full", s, 0.0005, True, .05)
275     time.sleep(1)
276
277 # go to rack
278 def go_to_rack(rack_number):
279     speed_z = 3000
280
281     if rack_number == 1:
282         run_motor("mx", 1, 300)
283         time.sleep(1)
284         run_motor("my", 1, 3400)
285         time.sleep(1)
286         run_motor("mz", 1, 2900)
287         time.sleep(1)
288         run_motor("my", 0, 400)
289         time.sleep(1)
290
291     elif rack_number == 2:
292         run_motor("mx", 1, 1600)
293         time.sleep(1)
294         run_motor("my", 1, 3400)
295         time.sleep(1)
296         run_motor("mz", 1, 3200)
297         time.sleep(1)
298         run_motor("my", 0, 400)
299         time.sleep(1)
300
20:37:29: This is Geany 1.37.1.
line: 179 / 481 col: 21 sel: 0 INS SP mode: LF encoding: UTF-8 filetype: Python scope: stop_motor

```

```

200     time.sleep(1)
201
202     #check motor x
203     INPUT_X = GPIO.input(switch_X)
204     if INPUT_X == 0:
205         run_motor("ax", 0, 4000)
206     time.sleep(1)
207
208 #run selected motor
209 def run_motor(m, d, s, set_point=None):
210
211     global motors
212     global switch_X
213     global switch_Y
214     global switch_Z
215     global motor_X_state
216     global motor_Y_state
217     global motor_Z_state
218
219     s = int(s)
220     d = True if int(d) == 1 else False
221
222     if m == "ax":
223         inp = GPIO.input(switch_X)
224         if inp == 1:
225             motor_X_state = 2
226         else:
227             motor_X_state = 1
228
229         motor = motors[m]["cls"]
230         motor.motor.go(d, "Full", s, .0005, True, .05)
231         time.sleep(1)
232
233     elif m == "my":
234         inp = GPIO.input(switch_Y)
235         if inp == 1:
236             motor_Y_state = 2
237         else:
238             motor_Y_state = 1
239
240         motor = motors[m]["cls"]
241         motor.motor.go(d, "Full", s, .0005, True, .05)
242         time.sleep(1)
243
244     elif m == "mz":
245         inp = GPIO.input(switch_Z)
246         if inp == 1:
247             motor_Z_state = 2
248         else:
249             motor_Z_state = 1
250
251         motor = motors[m]["cls"]
252         motor.motor.go(d, "Full", s, .0005, True, .05)
253         time.sleep(1)
254
255     set_starting_point()
256
257 #get available racks
258 def get_available_racks():
259
260     #available racks
261     racks = []
262     for rack in range(1, 6):
263         racks.append(rack)
264     return racks
265
266 #main
267 if __name__ == '__main__':
268     #get available racks
269     racks = get_available_racks()
270     #run selected motor
271     run_motor("ax", 0, 4000)
272     #run selected motor
273     run_motor("my", 0, 4000)
274     #run selected motor
275     run_motor("mz", 0, 4000)
276
277 #end
278

```

20:37:29: This is Geany 1.37.1.

line: 179 / 481 col: 21 sel: 0 INS SP mode: LF encoding: UTF-8 filetype: Python scope: stop\_motor

```

299     time.sleep(1)
300     run_motor("my", 0, 400)
301     time.sleep(1)
302
303     elif rack_number == 3:
304         run_motor("ax", 1, 2700)
305         time.sleep(1)
306         run_motor("my", 1, 3200)
307         time.sleep(1)
308         run_motor("ax", 1, 3000)
309         time.sleep(1)
310         run_motor("my", 0, 400)
311         time.sleep(1)
312
313     elif rack_number == 4:
314         run_motor("ax", 1, 300)
315         time.sleep(1)
316         run_motor("my", 1, 2200)
317         time.sleep(1)
318         run_motor("ax", 1, 3200)
319         time.sleep(1)
320         run_motor("my", 0, 400)
321         time.sleep(1)
322
323     elif rack_number == 5:
324         run_motor("ax", 1, 1600)
325         time.sleep(1)
326         run_motor("my", 1, 2200)
327         time.sleep(1)
328         run_motor("ax", 1, 4200)
329         time.sleep(1)
330         run_motor("my", 0, 400)
331         time.sleep(1)
332
333     elif rack_number == 6:
334         run_motor("ax", 1, 2700)
335         time.sleep(1)
336         run_motor("my", 1, 2200)
337         time.sleep(1)
338         run_motor("ax", 1, 3000)
339         time.sleep(1)
340         run_motor("my", 0, 400)
341         time.sleep(1)
342
343     set_starting_point()
344
345
346
347
348 #get available racks
349 def get_available_racks():
350
351     #available racks
352     racks = []
353     for rack in range(1, 6):
354         racks.append(rack)
355     return racks
356
357 #main
358 if __name__ == '__main__':
359     #get available racks
360     racks = get_available_racks()
361     #run selected motor
362     run_motor("ax", 0, 4000)
363     #run selected motor
364     run_motor("my", 0, 4000)
365     #run selected motor
366     run_motor("mz", 0, 4000)
367
368 #end
369

```

20:37:29: This is Geany 1.37.1.

line: 179 / 481 col: 21 sel: 0 INS SP mode: LF encoding: UTF-8 filetype: Python scope: stop\_motor

```

smartinventory.py - /home/raspberry/project/smartinventory - Geany
File Edit Search View Document Project Build Tools Help
Documents > requirements.txt x rfid.py x shape.py x smartinventory.py x shape_finder.py x motordc.py x motor.py x multimotor.py x button.py x
~/project
├── inventory.py
├── project...inventory
├── button.py
├── motor.py
├── motordc.py
├── multimotor.py
├── requirements.txt
├── rfid.py
├── shape.py
├── shape_finder.py
└── smartinventory.py
351 global used_racks
352
353 available_racks = [x for x in racks if x not in used_racks]
354 return available_racks
355
356 #register rack
357 def register_rack(card_id, rack_number):
358     global cards
359     global used_racks
360     used_racks.append(rack_number)
361     cards[card_id]["rack"] = rack_number
362     print(f"Rak tersedia: {used_racks}")
363
364     available_racks = get_available_racks()
365     print(f"Rak tersedia: {available_racks}")
366     print("=====")
367     return True
368
369 #check card
370 def check_card(card_id):
371     global cards
372     global max_item
373     #if card is registered
374     if card_id in cards:
375         print(f"Telah datang {card_id}")
376         print("=====")
377         time.sleep(2)
378     #if card has no rack registered
379     if cards[card_id]["rack"] is None:
380         available_racks = get_available_racks()
381         #if there is available rack
382         if len(available_racks) > 0:
383             my_rack = available_racks[0]
384             register_rack(card_id, my_rack)
385             # go to rack
386             go_to_rack(my_rack)
387             cards[card_id]["items"] += 1
388             return True
389         #if no available rack
390         else:
391             print(f"ERROR: Rak tidak tersedia")
392             return False
393     # if card has rack
394     else:
395         if cards[card_id]["items"] < max_item:
396             my_rack = cards[card_id]["rack"]
397             # go to rack
398             go_to_rack(my_rack)
399             cards[card_id]["items"] += 1
400             return True
401         else:
402             print(f"ERROR: Rak kartu {card_id} penuh")
403             return False
404     #if not registered
405     else:
406         print("ERROR: Kartu tidak terdaftar")
407         return False
408
409 # print verbose
410 def verbose(card_id):
411     global cards
412     print(f"No. Kartu: {card_id}")
413     print(f"No. Rak: {cards[card_id]['rack']}")
414     print(f"Total item: {cards[card_id]['items']}")
415     print("=====")
416
417 # initialize motors
418 init_motors()
419
420 try:
421     n = input("Tentukan maksimum item untuk tiap rak (default: 3): ")
422     n = int(n)
423
424     # initialize cards
425     cards = {}
426     for i in range(1, n+1):
427         cards[i] = {"rack": None, "items": 0}
428
429     # initialize racks
430     racks = []
431     for i in range(1, n+1):
432         racks.append(i)
433
434     # initialize used_racks
435     used_racks = []
436
437     # initialize available_racks
438     available_racks = []
439
440     # initialize max_item
441     max_item = n
442
443     # initialize verbose
444     verbose = False
445
446     # initialize verbose
447     verbose = False
448
449     # initialize verbose
450     verbose = False
451
452     # initialize verbose
453     verbose = False
454
455     # initialize verbose
456     verbose = False
457
458     # initialize verbose
459     verbose = False
460
461     # initialize verbose
462     verbose = False
463
464     # initialize verbose
465     verbose = False
466
467     # initialize verbose
468     verbose = False
469
470     # initialize verbose
471     verbose = False
472
473     # initialize verbose
474     verbose = False
475
476     # initialize verbose
477     verbose = False
478
479     # initialize verbose
480     verbose = False
481
482     # initialize verbose
483     verbose = False
484
485     # initialize verbose
486     verbose = False
487
488     # initialize verbose
489     verbose = False
490
491     # initialize verbose
492     verbose = False
493
494     # initialize verbose
495     verbose = False
496
497     # initialize verbose
498     verbose = False
499
500     # initialize verbose
501     verbose = False
502
503     # initialize verbose
504     verbose = False
505
506     # initialize verbose
507     verbose = False
508
509     # initialize verbose
510     verbose = False
511
512     # initialize verbose
513     verbose = False
514
515     # initialize verbose
516     verbose = False
517
518     # initialize verbose
519     verbose = False
520
521     # initialize verbose
522     verbose = False
523
524     # initialize verbose
525     verbose = False
526
527     # initialize verbose
528     verbose = False
529
530     # initialize verbose
531     verbose = False
532
533     # initialize verbose
534     verbose = False
535
536     # initialize verbose
537     verbose = False
538
539     # initialize verbose
540     verbose = False
541
542     # initialize verbose
543     verbose = False
544
545     # initialize verbose
546     verbose = False
547
548     # initialize verbose
549     verbose = False
550
551     # initialize verbose
552     verbose = False
553
554     # initialize verbose
555     verbose = False
556
557     # initialize verbose
558     verbose = False
559
560     # initialize verbose
561     verbose = False
562
563     # initialize verbose
564     verbose = False
565
566     # initialize verbose
567     verbose = False
568
569     # initialize verbose
570     verbose = False
571
572     # initialize verbose
573     verbose = False
574
575     # initialize verbose
576     verbose = False
577
578     # initialize verbose
579     verbose = False
580
581     # initialize verbose
582     verbose = False
583
584     # initialize verbose
585     verbose = False
586
587     # initialize verbose
588     verbose = False
589
590     # initialize verbose
591     verbose = False
592
593     # initialize verbose
594     verbose = False
595
596     # initialize verbose
597     verbose = False
598
599     # initialize verbose
600     verbose = False
601
602     # initialize verbose
603     verbose = False
604
605     # initialize verbose
606     verbose = False
607
608     # initialize verbose
609     verbose = False
610
611     # initialize verbose
612     verbose = False
613
614     # initialize verbose
615     verbose = False
616
617     # initialize verbose
618     verbose = False
619
620     # initialize verbose
621     verbose = False
622
623     # initialize verbose
624     verbose = False
625
626     # initialize verbose
627     verbose = False
628
629     # initialize verbose
630     verbose = False
631
632     # initialize verbose
633     verbose = False
634
635     # initialize verbose
636     verbose = False
637
638     # initialize verbose
639     verbose = False
640
641     # initialize verbose
642     verbose = False
643
644     # initialize verbose
645     verbose = False
646
647     # initialize verbose
648     verbose = False
649
650     # initialize verbose
651     verbose = False
652
653     # initialize verbose
654     verbose = False
655
656     # initialize verbose
657     verbose = False
658
659     # initialize verbose
660     verbose = False
661
662     # initialize verbose
663     verbose = False
664
665     # initialize verbose
666     verbose = False
667
668     # initialize verbose
669     verbose = False
670
671     # initialize verbose
672     verbose = False
673
674     # initialize verbose
675     verbose = False
676
677     # initialize verbose
678     verbose = False
679
680     # initialize verbose
681     verbose = False
682
683     # initialize verbose
684     verbose = False
685
686     # initialize verbose
687     verbose = False
688
689     # initialize verbose
690     verbose = False
691
692     # initialize verbose
693     verbose = False
694
695     # initialize verbose
696     verbose = False
697
698     # initialize verbose
699     verbose = False
700
701     # initialize verbose
702     verbose = False
703
704     # initialize verbose
705     verbose = False
706
707     # initialize verbose
708     verbose = False
709
710     # initialize verbose
711     verbose = False
712
713     # initialize verbose
714     verbose = False
715
716     # initialize verbose
717     verbose = False
718
719     # initialize verbose
720     verbose = False
721
722     # initialize verbose
723     verbose = False
724
725     # initialize verbose
726     verbose = False
727
728     # initialize verbose
729     verbose = False
730
731     # initialize verbose
732     verbose = False
733
734     # initialize verbose
735     verbose = False
736
737     # initialize verbose
738     verbose = False
739
740     # initialize verbose
741     verbose = False
742
743     # initialize verbose
744     verbose = False
745
746     # initialize verbose
747     verbose = False
748
749     # initialize verbose
750     verbose = False
751
752     # initialize verbose
753     verbose = False
754
755     # initialize verbose
756     verbose = False
757
758     # initialize verbose
759     verbose = False
760
761     # initialize verbose
762     verbose = False
763
764     # initialize verbose
765     verbose = False
766
767     # initialize verbose
768     verbose = False
769
770     # initialize verbose
771     verbose = False
772
773     # initialize verbose
774     verbose = False
775
776     # initialize verbose
777     verbose = False
778
779     # initialize verbose
780     verbose = False
781
782     # initialize verbose
783     verbose = False
784
785     # initialize verbose
786     verbose = False
787
788     # initialize verbose
789     verbose = False
790
791     # initialize verbose
792     verbose = False
793
794     # initialize verbose
795     verbose = False
796
797     # initialize verbose
798     verbose = False
799
800     # initialize verbose
801     verbose = False
802
803     # initialize verbose
804     verbose = False
805
806     # initialize verbose
807     verbose = False
808
809     # initialize verbose
810     verbose = False
811
812     # initialize verbose
813     verbose = False
814
815     # initialize verbose
816     verbose = False
817
818     # initialize verbose
819     verbose = False
820
821     # initialize verbose
822     verbose = False
823
824     # initialize verbose
825     verbose = False
826
827     # initialize verbose
828     verbose = False
829
830     # initialize verbose
831     verbose = False
832
833     # initialize verbose
834     verbose = False
835
836     # initialize verbose
837     verbose = False
838
839     # initialize verbose
840     verbose = False
841
842     # initialize verbose
843     verbose = False
844
845     # initialize verbose
846     verbose = False
847
848     # initialize verbose
849     verbose = False
850
851     # initialize verbose
852     verbose = False
853
854     # initialize verbose
855     verbose = False
856
857     # initialize verbose
858     verbose = False
859
860     # initialize verbose
861     verbose = False
862
863     # initialize verbose
864     verbose = False
865
866     # initialize verbose
867     verbose = False
868
869     # initialize verbose
870     verbose = False
871
872     # initialize verbose
873     verbose = False
874
875     # initialize verbose
876     verbose = False
877
878     # initialize verbose
879     verbose = False
880
881     # initialize verbose
882     verbose = False
883
884     # initialize verbose
885     verbose = False
886
887     # initialize verbose
888     verbose = False
889
890     # initialize verbose
891     verbose = False
892
893     # initialize verbose
894     verbose = False
895
896     # initialize verbose
897     verbose = False
898
899     # initialize verbose
900     verbose = False
901
902     # initialize verbose
903     verbose = False
904
905     # initialize verbose
906     verbose = False
907
908     # initialize verbose
909     verbose = False
910
911     # initialize verbose
912     verbose = False
913
914     # initialize verbose
915     verbose = False
916
917     # initialize verbose
918     verbose = False
919
920     # initialize verbose
921     verbose = False
922
923     # initialize verbose
924     verbose = False
925
926     # initialize verbose
927     verbose = False
928
929     # initialize verbose
930     verbose = False
931
932     # initialize verbose
933     verbose = False
934
935     # initialize verbose
936     verbose = False
937
938     # initialize verbose
939     verbose = False
940
941     # initialize verbose
942     verbose = False
943
944     # initialize verbose
945     verbose = False
946
947     # initialize verbose
948     verbose = False
949
950     # initialize verbose
951     verbose = False
952
953     # initialize verbose
954     verbose = False
955
956     # initialize verbose
957     verbose = False
958
959     # initialize verbose
960     verbose = False
961
962     # initialize verbose
963     verbose = False
964
965     # initialize verbose
966     verbose = False
967
968     # initialize verbose
969     verbose = False
970
971     # initialize verbose
972     verbose = False
973
974     # initialize verbose
975     verbose = False
976
977     # initialize verbose
978     verbose = False
979
980     # initialize verbose
981     verbose = False
982
983     # initialize verbose
984     verbose = False
985
986     # initialize verbose
987     verbose = False
988
989     # initialize verbose
990     verbose = False
991
992     # initialize verbose
993     verbose = False
994
995     # initialize verbose
996     verbose = False
997
998     # initialize verbose
999     verbose = False
1000

```

```

smartinventory.py - /home/raspberry/project/smartinventory - Geany
File Edit Search View Document Project Build Tools Help
Documents > requirements.txt x rfid.py x shape.py x smartinventory.py x shape_finder.py x motordc.py x motor.py x multimotor.py x button.py x
~/project
├── inventory.py
├── project...inventory
├── button.py
├── motor.py
├── motordc.py
├── multimotor.py
├── requirements.txt
├── rfid.py
├── shape.py
├── shape_finder.py
└── smartinventory.py
400 #if no available rack
401 else:
402     print(f"ERROR: Rak tidak tersedia")
403     return False
404
405 # if card has rack
406 else:
407     if cards[card_id]["items"] < max_item:
408         my_rack = cards[card_id]["rack"]
409         # go to rack
410         go_to_rack(my_rack)
411         cards[card_id]["items"] += 1
412         return True
413     else:
414         print(f"ERROR: Rak kartu {card_id} penuh")
415         return False
416
417 #if not registered
418 else:
419     print("ERROR: Kartu tidak terdaftar")
420     return False
421
422 # print verbose
423 def verbose(card_id):
424     global cards
425     print(f"No. Kartu: {card_id}")
426     print(f"No. Rak: {cards[card_id]['rack']}")
427     print(f"Total item: {cards[card_id]['items']}")
428     print("=====")
429
430 # initialize motors
431 init_motors()
432
433 try:
434     n = input("Tentukan maksimum item untuk tiap rak (default: 3): ")
435     n = int(n)
436
437     # initialize cards
438     cards = {}
439     for i in range(1, n+1):
440         cards[i] = {"rack": None, "items": 0}
441
442     # initialize racks
443     racks = []
444     for i in range(1, n+1):
445         racks.append(i)
446
447     # initialize used_racks
448     used_racks = []
449
450     # initialize available_racks
451     available_racks = []
452
453     # initialize max_item
454     max_item = n
455
456     # initialize verbose
457     verbose = False
458
459     # initialize verbose
460     verbose = False
461
462     # initialize verbose
463     verbose = False
464
465     # initialize verbose
466     verbose = False
467
468     # initialize verbose
469     verbose = False
470
471     # initialize verbose
472     verbose = False
473
474     # initialize verbose
475     verbose = False
476
477     # initialize verbose
478     verbose = False
479
480     # initialize verbose
481     verbose = False
482
483     # initialize verbose
484     verbose = False
485
486     # initialize verbose
487     verbose = False
488
489     # initialize verbose
490     verbose = False
491
492     # initialize verbose
493     verbose = False
494
495     # initialize verbose
496     verbose = False
497
498     # initialize verbose
499     verbose = False
500
501     # initialize verbose
502     verbose = False
503
504     # initialize verbose
505     verbose = False
506
507     # initialize verbose
508     verbose = False
509
510     # initialize verbose
511     verbose = False
512
513     # initialize verbose
514     verbose = False
515
516     # initialize verbose
517     verbose = False
518
519     # initialize verbose
520     verbose = False
521
522     # initialize verbose
523     verbose = False
524
525     # initialize verbose
526     verbose = False
527
528     # initialize verbose
529     verbose = False
530
531     # initialize verbose
532     verbose = False
533
534     # initialize verbose
535     verbose = False
536
537     # initialize verbose
538     verbose = False
539
540     # initialize verbose
541     verbose = False
542
543     # initialize verbose
544     verbose = False
545
546     # initialize verbose
547     verbose = False
548
549     # initialize verbose
550     verbose = False
551
552     # initialize verbose
553     verbose = False
554
555     # initialize verbose
556     verbose = False
557
558     # initialize verbose
559     verbose = False
560
561     # initialize verbose
562     verbose = False
563
564     # initialize verbose
565     verbose = False
566
567     # initialize verbose
568     verbose = False
569
570     # initialize verbose
571     verbose = False
572
573     # initialize verbose
574     verbose = False
575
576     # initialize verbose
577     verbose = False
578
579     # initialize verbose
580     verbose = False
581
582     # initialize verbose
583     verbose = False
584
585     # initialize verbose
586     verbose = False
587
588     # initialize verbose
589     verbose = False
590
591     # initialize verbose
592     verbose = False
593
594     # initialize verbose
595     verbose = False
596
597     # initialize verbose
598     verbose = False
599
600     # initialize verbose
601     verbose = False
602
603     # initialize verbose
604     verbose = False
605
606     # initialize verbose
607     verbose = False
608
609     # initialize verbose
610     verbose = False
611
612     # initialize verbose
613     verbose = False
614
615     # initialize verbose
616     verbose = False
617
618     # initialize verbose
619     verbose = False
620
621     # initialize verbose
622     verbose = False
623
624     # initialize verbose
625     verbose = False
626
627     # initialize verbose
628     verbose = False
629
630     # initialize verbose
631     verbose = False
632
633     # initialize verbose
634     verbose = False
635
636     # initialize verbose
637     verbose = False
638
639     # initialize verbose
640     verbose = False
641
642     # initialize verbose
643     verbose = False
644
645     # initialize verbose
646     verbose = False
647
648     # initialize verbose
649     verbose = False
650
651     # initialize verbose
652     verbose = False
653
654     # initialize verbose
655     verbose = False
656
657     # initialize verbose
658     verbose = False
659
660     # initialize verbose
661     verbose = False
662
663     # initialize verbose
664     verbose = False
665
666     # initialize verbose
667     verbose = False
668
669     # initialize verbose
670     verbose = False
671
672     # initialize verbose
673     verbose = False
674
675     # initialize verbose
676     verbose = False
677
678     # initialize verbose
679     verbose = False
680
681     # initialize verbose
682     verbose = False
683
684     # initialize verbose
685     verbose = False
686
687     # initialize verbose
688     verbose = False
689
690     # initialize verbose
691     verbose = False
692
693     # initialize verbose
694     verbose = False
695
696     # initialize verbose
697     verbose = False
698
699     # initialize verbose
700     verbose = False
701
702     # initialize verbose
703     verbose = False
704
705     # initialize verbose
706     verbose = False
707
708     # initialize verbose
709     verbose = False
710
711     # initialize verbose
712     verbose = False
713
714     # initialize verbose
715     verbose = False
716
717     # initialize verbose
718     verbose = False
719
720     # initialize verbose
721     verbose = False
722
723     # initialize verbose
724     verbose = False
725
726     # initialize verbose
727     verbose = False
728
729     # initialize verbose
730     verbose = False
731
732     # initialize verbose
733     verbose = False
734
735     # initialize verbose
736     verbose = False
737
738     # initialize verbose
739     verbose = False
740
741     # initialize verbose
742     verbose = False
743
744     # initialize verbose
745     verbose = False
746
747     # initialize verbose
748     verbose = False
749
750     # initialize verbose
751     verbose = False
752
753     # initialize verbose
754     verbose = False
755
756     # initialize verbose
757     verbose = False
758
759     # initialize verbose
760     verbose = False
761
762     # initialize verbose
763     verbose = False
764
765     # initialize verbose
766     verbose = False
767
768     # initialize verbose
769     verbose = False
770
771     # initialize verbose
772     verbose = False
773
774     # initialize verbose
775     verbose = False
776
777     # initialize verbose
778     verbose = False
779
780     # initialize verbose
781     verbose = False
782
783     # initialize verbose
784     verbose = False
785
786     # initialize verbose
787     verbose = False
788
789     # initialize verbose
790     verbose = False
791
792     # initialize verbose
793     verbose = False
794
795     # initialize verbose
796     verbose = False
797
798     # initialize verbose
799     verbose = False
800
801     # initialize verbose
802     verbose = False
803
804     # initialize verbose
805     verbose = False
806
807     # initialize verbose
808     verbose = False
809
810     # initialize verbose
811     verbose = False
812
813     # initialize verbose
814     verbose = False
815
816     # initialize verbose
817     verbose = False
818
819     # initialize verbose
820     verbose = False
821
822     # initialize verbose
823     verbose = False
824
825     # initialize verbose
826     verbose = False
827
828     # initialize verbose
829     verbose = False
830
831     # initialize verbose
832     verbose = False
833
834     # initialize verbose
835     verbose = False
836
837     # initialize verbose
838     verbose = False
839
840     # initialize verbose
841     verbose = False
842
843     # initialize verbose
844     verbose = False
845
846     # initialize verbose
847     verbose = False
848
849     # initialize verbose
850     verbose = False
851
852     # initialize verbose
853     verbose = False
854
855     # initialize verbose
856     verbose = False
857
858     # initialize verbose
859     verbose = False
860
861     # initialize verbose
862     verbose = False
863
864     # initialize verbose
865     verbose = False
866
867     # initialize verbose
868     verbose = False
869
870     # initialize verbose
871     verbose = False
872
873     # initialize verbose
874     verbose = False
875
876     # initialize verbose
877     verbose = False
878
879     # initialize verbose
880     verbose = False
881
882     # initialize verbose
883     verbose = False
884
885     # initialize verbose
886     verbose = False
887
888     # initialize verbose
889     verbose = False
890
891     # initialize verbose
892     verbose = False
893
894     # initialize verbose
895     verbose = False
896
897     # initialize verbose
898     verbose = False
899
900     # initialize verbose
901     verbose = False
902
903     # initialize verbose
904     verbose = False
905
906     # initialize verbose
907     verbose = False
908
909     # initialize verbose
910     verbose = False
911
912     # initialize verbose
913     verbose = False
914
915     # initialize verbose
916     verbose = False
917
918     # initialize verbose
919     verbose = False
920
921     # initialize verbose
922     verbose = False
923
924     # initialize verbose
925     verbose = False
926
927     # initialize verbose
928     verbose = False
929
930     # initialize verbose
931     verbose = False
932
933     # initialize verbose
934     verbose = False
935
936     # initialize verbose
937     verbose = False
938
939     # initialize verbose
940     verbose = False
941
942     # initialize verbose
943     verbose = False
944
945     # initialize verbose
946     verbose = False
947
948     # initialize verbose
949     verbose = False
950
951     # initialize verbose
952     verbose = False
953
954     # initialize verbose
955     verbose = False
956
957     # initialize verbose
958     verbose = False
959
960     # initialize verbose
961     verbose = False
962
963     # initialize verbose
964     verbose = False
965
966     # initialize verbose
967     verbose = False
968
969     # initialize verbose
970     verbose = False
971
972     # initialize verbose
973     verbose = False
974
975     # initialize verbose
976     verbose = False
977
978     # initialize verbose
979     verbose = False
980
981     # initialize verbose
982     verbose = False
983
984     # initialize verbose
985     verbose = False
986
987     # initialize verbose
988     verbose = False
989
990     # initialize verbose
991     verbose = False
992
993     # initialize verbose
994     verbose = False
995
996     # initialize verbose
997     verbose = False
998
999     # initialize verbose
1000     verbose = False

```

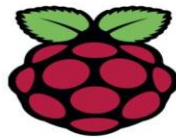
```

smartinventory.py - /home/raspberry/project/smartinventory - Geany
File Edit Search View Document Project Build Tools Help
Documents > requirements.txt x rfid.py x shape.py x smartinventory.py x shape_finder.py x motordc.py x motor.py x multimotor.py x button.py x
- /project
  - inventory.py
  - project...inventory
    - button.py
    - motor.py
    - motordc.py
    - multimotor.py
    - requirements.txt
    - rfid.py
    - shape.py
    - shape_finder.py
    - smartinventory.py
432 # print verbose
433
434 def verbose(card_id):
435     global cards
436
437     print("No. Kartu: {card_id}")
438     print("No. Rak: {cards[card_id]['rack']}")
439     print("Total item: {cards[card_id]['items']}")
440     print("=====")
441
442
443 # initialize motors
444 init_motors()
445
446
447 try:
448     n = input("Tentukan maksimum item untuk tiap rak (default: 3): ")
449
450     if n:
451         max_item = n
452
453     while(1):
454         print("Scan kartu member....")
455         id, text = reader.read()
456         #print(id, text)
457
458         card_id = str(id)
459         beep()
460         valid = check_card(card_id)
461
462         if valid:
463             beep(2)
464             verbose(card_id)
465         else:
466             beep(3)
467
468         #command = input("Masukkan perintah, contoh: m,d,s: ")
469         #m, d, s = command.split(",")
470         #run_motor(m, d, s)
471         #time.sleep(1)
472
473
474 finally:
475     tearDown()
476
477
478
479
480
481
20:37:29: This is Geany 1.37.1.
line: 179 / 481 col: 21 sel: 0 INS SP mode: LF encoding: UTF-8 filetype: Python scope: stop_motor

```

## Data Sheet Raspberry Pi 3

### DATASHEET



## Raspberry Pi 4 Model B

Release 1

June 2019

Copyright 2019 Raspberry Pi (Trading) Ltd. All rights reserved.







Symbol	Parameter	Minimum	Maximum	Unit
VIN	5V Input Voltage	-0.5	6.0	V

Table 2: Absolute Maximum Ratings

Please note that VDD.IO is the GPIO bank voltage which is tied to the on-board 3.3V supply rail.

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
V <sub>IL</sub>	Input low voltage <sup>a</sup>	VDD.IO = 3.3V	-	-	TBD	V
V <sub>IH</sub>	Input high voltage <sup>a</sup>	VDD.IO = 3.3V	TBD	-	-	V
I <sub>IL</sub>	Input leakage current	TA = +85°C	-	-	TBD	µA
C <sub>IN</sub>	Input capacitance	-	-	TBD	-	pF
V <sub>OL</sub>	Output low voltage <sup>b</sup>	VDD.IO = 3.3V, IOL = -2mA	-	-	TBD	V
V <sub>OIH</sub>	Output high voltage <sup>b</sup>	VDD.IO = 3.3V, IOH = 2mA	TBD	-	-	V
I <sub>OL</sub>	Output low current <sup>c</sup>	VDD.IO = 3.3V, VO = 0.4V	TBD	-	-	mA
I <sub>OIH</sub>	Output high current <sup>c</sup>	VDD.IO = 3.3V, VO = 2.3V	TBD	-	-	mA
R <sub>Pu</sub>	Pullup resistor	-	TBD	-	TBD	kΩ
R <sub>Pd</sub>	Pulldown resistor	-	TBD	-	TBD	kΩ

<sup>a</sup> Hysteresis enabled  
<sup>b</sup> Default drive strength (8mA)  
<sup>c</sup> Maximum drive strength (16mA)

Table 3: DC Characteristics

Pin Name	Symbol	Parameter	Minimum	Typical	Maximum	Unit
Digital outputs	t <sub>rise</sub>	10-90% rise time <sup>a</sup>	-	TBD	-	ns
Digital outputs	t <sub>fall</sub>	90-10% fall time <sup>a</sup>	-	TBD	-	ns

<sup>a</sup> Default drive strength, CL = 5pF, VDD.IO = 3.3V

Table 4: Digital I/O Pin AC Characteristics



Figure 2: Digital IO Characteristics



### 4.1 Power Requirements

The Pi4B requires a good quality USB-C power supply capable of delivering 5V at 3A. If attached downstream USB devices consume less than 500mA, a 5V, 2.5A supply may be used.

## 5 Peripherals

### 5.1 GPIO Interface

The Pi4B makes 28 BCM2711 GPIOs available via a standard Raspberry Pi 40-pin header. This header is backwards compatible with all previous Raspberry Pi boards with a 40-way header.

#### 5.1.1 GPIO Pin Assignments

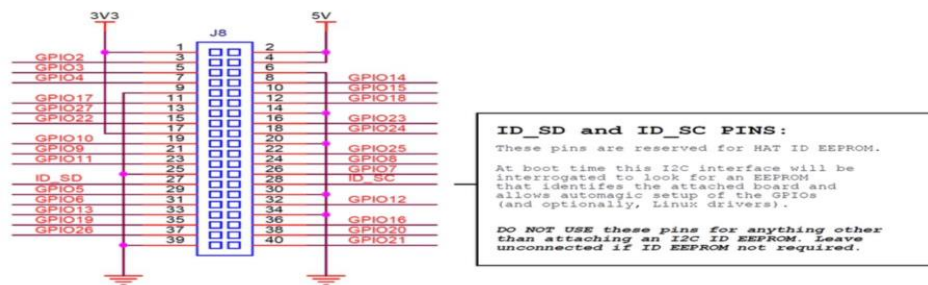


Figure 3: GPIO Connector Pinout

As well as being able to be used as straightforward software controlled input and output (with programmable pulls), GPIO pins can be switched (multiplexed) into various other modes backed by dedicated peripheral blocks such as I2C, UART and SPI.

In addition to the standard peripheral options found on legacy Pis, extra I2C, UART and SPI peripherals have been added to the BCM2711 chip and are available as further mux options on the Pi4. This gives users much more flexibility when attaching add-on hardware as compared to older models.



### 5.1.2 GPIO Alternate Functions

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
0	High	SDA0	SA5	PCLK	SPI3_CEO_N	TXD2	SDA6
1	High	SCL0	SA4	DE	SPI3_MISO	RXD2	SCL6
2	High	SDA1	SA3	LCD_VSYNC	SPI3_MOSI	CTS2	SDA3
3	High	SCL1	SA2	LCD_HSYNC	SPI3_SCLK	RTS2	SCL3
4	High	GPCLK0	SA1	DPLD0	SPI4_CEO_N	TXD3	SDA3
5	High	GPCLK1	SA0	DPLD1	SPI4_MISO	RXD3	SCL3
6	High	GPCLK2	SOE_N	DPLD2	SPI4_MOSI	CTS3	SDA4
7	High	SPI0_CEO_N	SWE_N	DPLD3	SPI4_SCLK	RTS3	SCL4
8	High	SPI0_CEO_N	SD0	DPLD4	-	TXD4	SDA4
9	Low	SPI0_MISO	SD1	DPLD5	-	RXD4	SCL4
10	Low	SPI0_MOSI	SD2	DPLD6	-	CTS4	SDA5
11	Low	SPI0_SCLK	SD3	DPLD7	-	RTS4	SCL5
12	Low	PWM0	SD4	DPLD8	SPI5_CEO_N	TXD5	SDA5
13	Low	PWM1	SD5	DPLD9	SPI5_MISO	RXD5	SCL5
14	Low	TXD0	SD6	DPLD10	SPI5_MOSI	CTS5	TXD1
15	Low	RXD0	SD7	DPLD11	SPI5_SCLK	RTS5	RXD1
16	Low	FL0	SD8	DPLD12	CTS0	SPI1_CEO_N	CTS1
17	Low	FL1	SD9	DPLD13	RTS0	SPI1_CEO_N	RTS1
18	Low	PCM_CLK	SD10	DPLD14	SPI6_CEO_N	SPI1_CEO_N	PWM0
19	Low	PCM_FS	SD11	DPLD15	SPI6_MISO	SPI1_MISO	PWM1
20	Low	PCM_DIN	SD12	DPLD16	SPI6_MOSI	SPI1_MOSI	GPCLK0
21	Low	PCM_DOUT	SD13	DPLD17	SPI6_SCLK	SPI1_SCLK	GPCLK1
22	Low	SD0_CLK	SD14	DPLD18	SD1_CLK	ARM_TRST	SDA6
23	Low	SD0_CMD	SD15	DPLD19	SD1_CMD	ARM_RTCK	SCL6
24	Low	SD0_DAT0	SD16	DPLD20	SD1_DAT0	ARM_TDO	SPI3_CEO_N
25	Low	SD0_DAT1	SD17	DPLD21	SD1_DAT1	ARM_TCK	SPI4_CEO_N
26	Low	SD0_DAT2	TE0	DPLD22	SD1_DAT2	ARM_TDI	SPI5_CEO_N
27	Low	SD0_DAT3	TE1	DPLD23	SD1_DAT3	ARM_TMS	SPI6_CEO_N

Table 5: Raspberry Pi 4 GPIO Alternate Functions

Table 5 details the default pin pull state and available alternate GPIO functions. Most of these alternate peripheral functions are described in detail in the BCM2711 Peripherals Specification document which can be downloaded from the [hardware documentation](#) section of the website.



### 5.1.3 Display Parallel Interface (DPI)

A standard parallel RGB (DPI) interface is available the GPIOs. This up-to-24-bit parallel interface can support a secondary display.

### 5.1.4 SD/SDIO Interface

The Pi4B has a dedicated SD card socket which supports 1.8V, DDR50 mode (at a peak bandwidth of 50 Megabytes / sec). In addition, a legacy SDIO interface is available on the GPIO pins.

## 5.2 Camera and Display Interfaces

The Pi4B has 1x Raspberry Pi 2-lane MIPI CSI Camera and 1x Raspberry Pi 2-lane MIPI DSI Display connector. These connectors are backwards compatible with legacy Raspberry Pi boards, and support all of the available Raspberry Pi camera and display peripherals.

## 5.3 USB

The Pi4B has 2x USB2 and 2x USB3 type-A sockets. Downstream USB current is limited to approximately 1.1A in aggregate over the four sockets.

## 5.4 HDMI

The Pi4B has 2x micro-HDMI ports, both of which support CEC and HDMI 2.0 with resolutions up to 4Kp60.

## 5.5 Audio and Composite (TV Out)

The Pi4B supports near-CD-quality analogue audio output and composite TV-output via a 4-ring TRS 'A/V' jack.

The analog audio output can drive 32 Ohm headphones directly.

## 5.6 Temperature Range and Thermals

The recommended ambient operating temperature range is 0 to 50 degrees Celcius.

To reduce thermal output when idling or under light load, the Pi4B reduces the CPU clock speed and voltage. During heavier load the speed and voltage (and hence thermal output) are increased. The internal governor will throttle back both the CPU speed and voltage to make sure the CPU temperature never exceeds 85 degrees C.

The Pi4B will operate perfectly well without any extra cooling and is designed for sprint performance - expecting a light use case on average and ramping up the CPU speed when needed (e.g. when loading a webpage). If a user wishes to load the system continually or operate it at a high temperature at full performance, further cooling may be needed.

## Data Sheet Driver Motor A4988



A4988

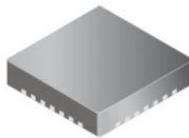
## DMOS Microstepping Driver with Translator and Overcurrent Protection

### Features and Benefits

- Low  $R_{DS(ON)}$  outputs
- Automatic current decay mode detection/selection
- Mixed and Slow current decay modes
- Synchronous rectification for low power dissipation
- Internal UVLO
- Crossover-current protection
- 3.3 and 5 V compatible logic supply
- Thermal shutdown circuitry
- Short-to-ground protection
- Shorted load protection
- Five selectable step modes: full,  $1/2$ ,  $1/4$ ,  $1/8$ , and  $1/16$

### Package:

28-contact QFN  
with exposed thermal pad  
5 mm × 5 mm × 0.90 mm  
(ET package)



Approximate size

### Description

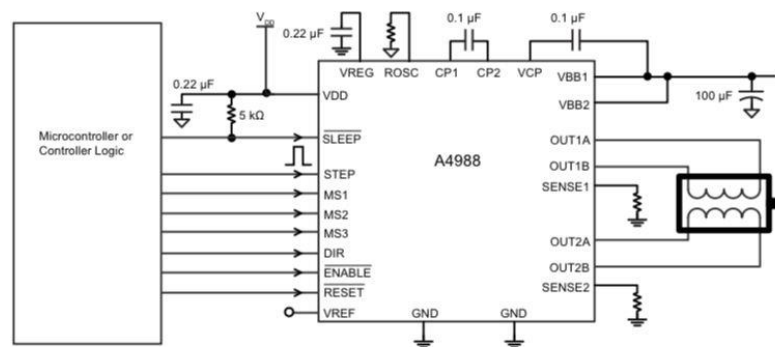
The A4988 is a complete microstepping motor driver with built-in translator for easy operation. It is designed to operate bipolar stepper motors in full-, half-, quarter-, eighth-, and sixteenth-step modes, with an output drive capacity of up to 35 V and  $\pm 2$  A. The A4988 includes a fixed off-time current regulator which has the ability to operate in Slow or Mixed decay modes.

The translator is the key to the easy implementation of the A4988. Simply inputting one pulse on the STEP input drives the motor one microstep. There are no phase sequence tables, high frequency control lines, or complex interfaces to program. The A4988 interface is an ideal fit for applications where a complex microprocessor is unavailable or is overburdened.

During stepping operation, the chopping control in the A4988 automatically selects the current decay mode, Slow or Mixed. In Mixed decay mode, the device is set initially to a fast decay for a proportion of the fixed off-time, then to a slow decay for the remainder of the off-time. Mixed decay current control results in reduced audible motor noise, increased step accuracy, and reduced power dissipation.

*Continued on the next page...*

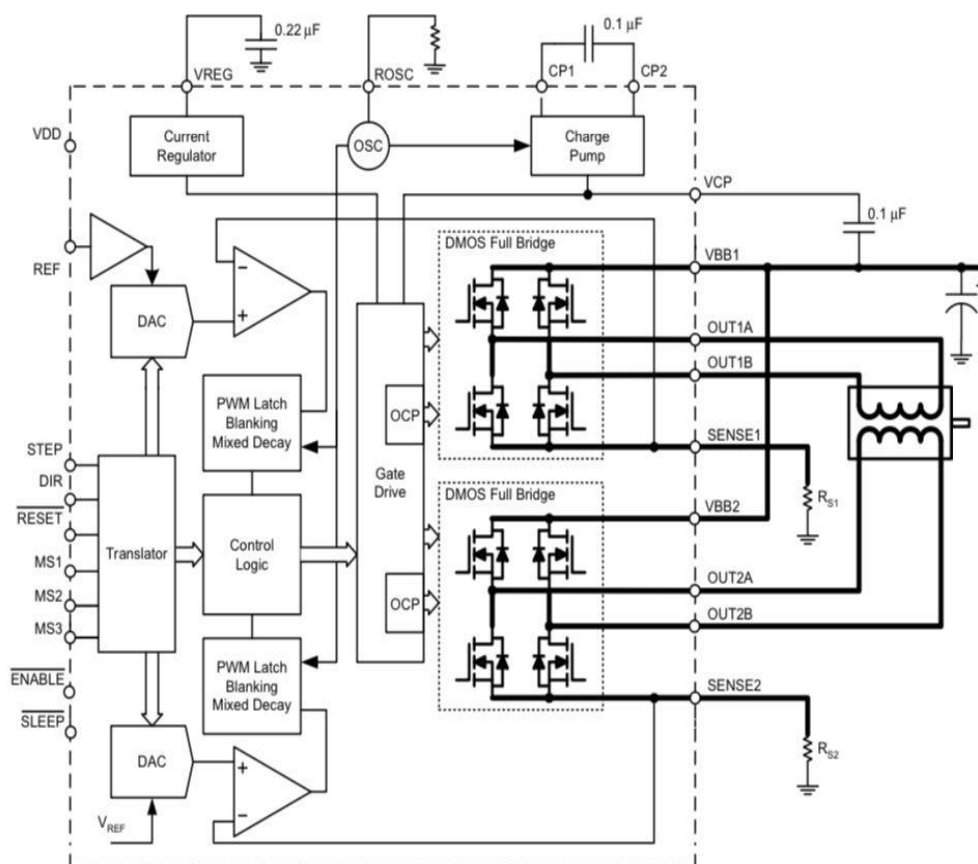
### Typical Application Diagram



## A4988

# DMOS Microstepping Driver with Translator and Overcurrent Protection

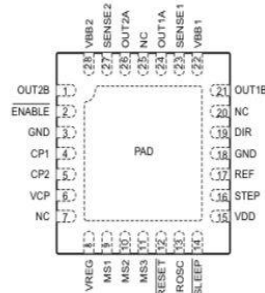
Functional Block Diagram



A4988

## DMOS Microstepping Driver with Translator and Overcurrent Protection

Pin-out Diagram



Terminal List Table

Name	Number	Description
CP1	4	Charge pump capacitor terminal
CP2	5	Charge pump capacitor terminal
VCP	6	Reservoir capacitor terminal
VREG	8	Regulator decoupling terminal
MS1	9	Logic input
MS2	10	Logic input
MS3	11	Logic input
RESET	12	Logic input
ROSC	13	Timing set
SLEEP	14	Logic input
VDD	15	Logic supply
STEP	16	Logic input
REF	17	$G_m$ reference voltage input
GND	3, 18	Ground*
DIR	19	Logic input
OUT1B	21	DMOS Full Bridge 1 Output B
VBB1	22	Load supply
SENSE1	23	Sense resistor terminal for Bridge 1
OUT1A	24	DMOS Full Bridge 1 Output A
OUT2A	26	DMOS Full Bridge 2 Output A
SENSE2	27	Sense resistor terminal for Bridge 2
VBB2	28	Load supply
OUT2B	1	DMOS Full Bridge 2 Output B
ENABLE	2	Logic input
NC	7, 20, 25	No connection
PAD	–	Exposed pad for enhanced thermal dissipation*

\*The GND pins must be tied together externally by connecting to the PAD ground plane under the device.

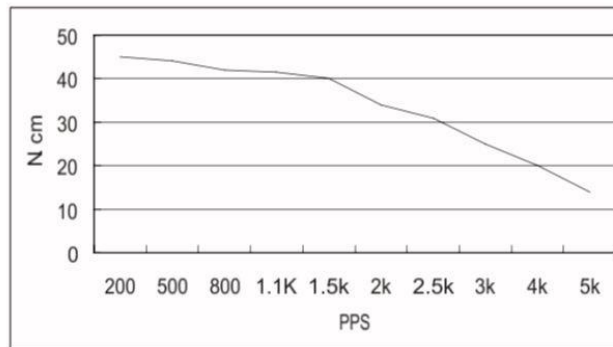
**Data Sheet Motor Stepper NEMA 17**

**HIGH TORQUE HYBRID STEPPING MOTOR SPECIFICATIONS**

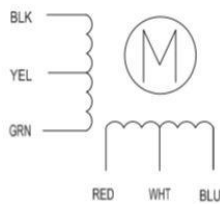
General specifications		Electrical specifications	
Step Angle (°)	1.8	Rated Voltage (V)	4
Temperature Rise (°C)	80 Max (rated current, 2 phase on)	Rated Current (A)	1.2
Ambient temperature (°C)	-20~+50	Resistance Per Phase ( $\pm 10\% \Omega$ )	3.3 (25°C)
Number of Phase	2	Inductance Per Phase ( $\pm 20\% \text{mH}$ )	2.8
Insulation Resistance	100M $\Omega$ , Min (500VDC)	Holding Torque (Kg.cm)	3.17
Insulation Class	Class B	Detent Torque (g.cm)	200
Max. radial force (N)	28 (20mm from the flange)	Rotor Inertia (g.cm <sup>2</sup> )	68
Max. axial force (N)	10	Weight (Kg)	0.365

● Pull out torque curve:

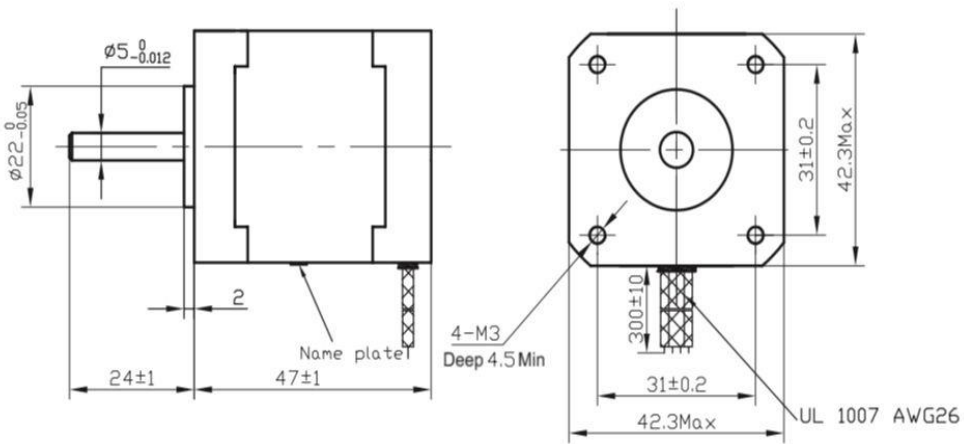
VOLTAGE: 24VDC, CONSTANT CURRENT: 1.2A, HALF STEP



● Wiring Diagram:



● Dimensions: (unit=mm)



SY42STH47-1206A				TECHNICAL CONDITIONS	
REV	REVISIONS	DESCRIPTION	BY	DATE	060047000
DRAW	任飞飞	2010.06.29			
CHECK					
APPROVE					
CHANGZHOU SONGYANG MACHINERY & ELECTRONICS NEW TECHNIC INSTITUTE					

## Data Sheet Sensor RFID



### RFID PROXIMITY ACCESS CONTROL

### RFID231C-SYS

#### Features

- Complete Proximity Access System Kit
- Transponder Cards Read Up To 4 Inches Range With Internal Aerial.
- System Includes
  - Control & Sensor Units
  - 10 Transponder Cards.
  - Power Supply Unit
  - Connecting Cable
  - Door Strike
  - All Screws & Fixings
- Exit Switch Option
- Programmable up to 50 Cards.
- Led Indicators For Power & Accept/Reject.
- Relay Contacts 2A@12Vdc.
- User Programmable Relay, 0.5 Secs To 10 Mins
- Easy Learn, Erase Cards Via Internal Switch.
- RS232 Serial Coms to PC.
- Real Time Data Logging Output.



#### Description

The RFID Proximity Access System provides controlled access to premises by means of high-security transponder card technology. These are similar to credit cards in size and appearance, but no physical contact is required with the reader for the card to be scanned.

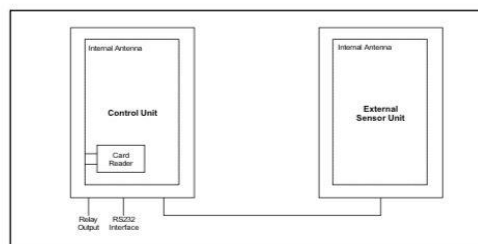
When an authorised card is placed within 4 inches of the sensor, the relay output will operate for a preset time. The system consists of a control unit and a remote sensor unit that can be fitted to the outside of a building. The control unit simply requires power and relay connections to operate. The sensor unit connects to the control unit with the cable provided.

The system has the capacity to learn up to 50 transponder cards. Programming is achieved using either a pushbutton inside the control unit, or from PC connected to the Control Unit RS232 serial port (only two connections required).

Connection to a PC enables many other functions including; control of the relay output from the PC, data logging information of which transponder card has operated the system.

The system is supplied in an ABS plastic enclosures and are IP65 rated.

#### Block diagram





## RFID PROXIMITY ACCESS CONTROL

### RFID231C-SYS

#### System Installation

A full installation guide is provided with the system (DS000230). This is also available from our website.

Technical specification		
Features	Specification	Comments
Protocol	Hitag2	Philips high security Protocol
User Range	Upto 15cm	Using standard Transponder card
Maximum No of Cards	50	
User feedback	LED	Data Accept / Reject LED Indication on Sensor and Control Unit
Programming Features	Yes	Via 2 onboard Switches or PC control Interface
Relay output	2A @ 12Vdc	Standard British Telecom BT47W/6 Changeover Contacts
Relay Operation Time	½ Sec – 10 mins	Programmable via onboard Switches or PC
PC Interface	RS232	2 wires only!
PC Commands	9	Complete Configuration through PC is possible.
Data Logging Output	Yes	Real time through RS232
Mounting	Wall or Pole	With Screw fixings provided
Environmental Protection	IP65	
Enclosure Material	ABS	
Power Supply: UK Version	UK 230Vac	UK Plug top
Power Supply: INTL Version	INTL 110 –240Vac	Standard International IEC Input
Power Consumption	~50mA	Quiescent
	~90mA	Operating Relay
Dimensions	110 x 85 x 35mm	
Operating Temp	-0 to + 70 °C	
Storage Temp	-55 to + 125 °C	

Technical specification: Transponder Card		
Features	Specification	Comments
Card Type	ISO Hitag2	Can be any Hitag2 card package
Material	White PVC	May be Custom Silk Screened
Size		As standard credit card

PART No	DESCRIPTION
RFID231C-SYS	RFID Proximity Access System with 10 Cards
RFCARDHITAG2	Spare RFID Card tag (Pack of 10)

Should you require further assistance, please call;

**R. F. Solutions Ltd,**  
Unit 21, Cliffe Ind Est, South St, Lewes, E Sussex, BN8 6JL. England.

Tel +44 (0)1273 898 000.

Fax +44 (0)1273 480 661.

Email [sales@rfsolutions.co.uk](mailto:sales@rfsolutions.co.uk)

<http://www.rfsolutions.co.uk>

RF Solutions is a member of the Low Power Radio Association.



Information contained in this document is believed to be accurate, however no representation or warranty is given and no liability is assumed by R.F. Solutions Ltd. with respect to the accuracy of such information. Use of R.F.Solutions as critical components in life support systems is not authorised except with express written approval from R.F. Solutions Ltd.





ISSN 2477-2755 (P) / 2622-2981 (E)

**SURAT KETERANGAN PENERIMAAN NASKAH  
No. 33/Ampere/VIII/2023**

Dewan Redaksi Jurnal Ampere, menyatakan bahwa:

Judul **Implementasi DOF (Degree Of Freedom) Pada Pergerakan Motor Stepper Smart Inventory 3 Axis**  
Penulis **Ali Ripaldo1, Yordan Hasan2\*, Johansyah Al Rasyid3**  
Afiliasi **Prodi Teknik Elektro Mekatronika, Jurusan Teknik Elektro, Politeknik Negeri Sriwijaya, Indonesia**

Adalah benar telah mengirimkan artikelnya ke Jurnal Ampere dan saat ini sedang dalam proses **PEER REVIEW** sesuai dengan prosedur Jurnal Ampere Universitas PGRI Palembang,

Demikian Surat Keterangan ini dibuat, untuk dapat dipergunakan sebagaimana perlunya.



Palembang, 7 Agustus 2022

Redaksi Jurnal Ampere

  
Nita Nurdiana, MT



**Jurnal Ampere**  
Prodi Teknik Elektro Fakultas Teknik Universitas PGRI Palembang  
Jl. A. Yani Lrg. Gotong Royong 9-10 Ulu Palembang  
Email : [ampereupgrip@gmail.com](mailto:ampereupgrip@gmail.com)