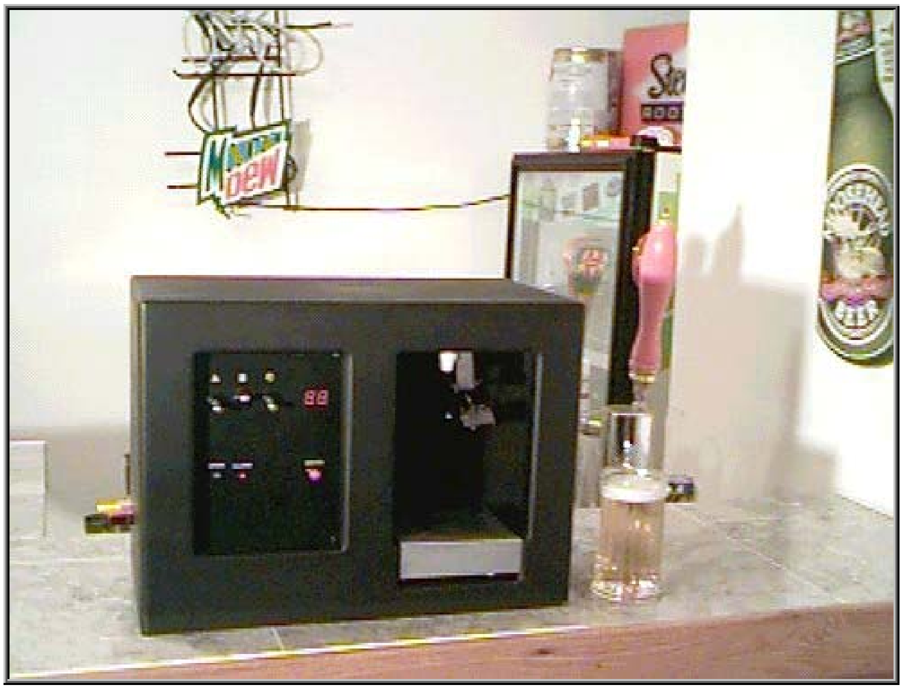
Gadget Freak

The BeerBot is the Next Generation in Beer Security

Design News
April 26, 2004



To keep his roommates from pilfering his beer reserves, Ryan resolved to build a Fort Knox-like device to protect his brew. The BeerBot waits for a correct activation code and then pours beer until a sensor detects that the cup is full. If a wrong code is entered, a speaker sounds an alarm and a counter is incremented—a combo that would surely prevent all but the most foolhardy from attempting another break-in. Ryan now sleeps peacefully knowing his beer is safe and sound.

BeerBot parts list

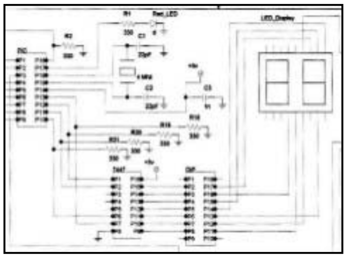
Amt	Part Description	Allied Part #
1	7-segment LED display	505-5462
1	Green LED	263-1239
1	Red LED	263-0115
1	330-Ohm DIP resistor array	755-4815
1	Piezoelectric speaker	854-6636
1	AND gate (quad 2-input)	263-2778
1	Inverter (hex)	263-0167
1	BCD to 7-segment LED decoder	735-4097
1	4 MHz Crystal	614-0014
1	SPDT Relay	686-0013
1	1N4001 Diode	266-0001
2	2N3904 npn transistor	568-8253
1	Push button (NO momentary)	855-1065
3	Toggle switch (SPST)	683-0048
1	Microswitch	676-4198
1	5V Regulator	568-3101
Additional parts required: Capacitors, resistors, PIC microcontroller, old CD-ROM drive, fluid pump and tubing, fluid sensor, some brewskis		

Description:

The BeerBot requires a code to be entered before it will operate. If the three toggle switches are set to an incorrect combination and the enter button is pushed, the 7-segment LED increments one and sounds an alarm. Each additional time an incorrect combination is entered, the display counts up one and an increasingly annoying alarm is sounded. Once the correct combination is entered, a tray is ejected and waits for a cup to be placed on it (detected by a micro switch). The tray then retracts and, as it hits a hinge on its way back into the housing, a fluid detector is lowered into the cup. A pump then begins to pour hoppy goodness from a reservoir into the cup until either the enter button is pressed again or the cup is full.

Electrical:

One of the motivations for designing the BeerBot (besides thwarting my thieving roommates) was to create a project to explore the functionality of a PIC microcontroller, as well as other basic circuit elements. So, this is a good beginners' project to start playing with microcontrollers and circuits in general. I used a PIC16F84, though there a lot of other PIC models that would work equally well. The PIC16F84 has only 13 I/Os, so some additional components were used to aid in the logic. For example, AND gates and inverters were used to reduce the toggle combination to a single input and a BCD to 7-segment LED decoder is used to reduce the 7-segment display outputs from seven to four.



(Click for full wiring diagram)

As a first foray into PIC programming, it is extremely advantageous to use a high level programming language like PicBasic Pro. Assembly language is the main alternative and likely to turn you off from microcontrollers if you're a newbie to writing code. Here is my PicBasic code for the BeerBot. Even if you've never seen PicBasic, it is a very readable language and should let you better understand how the switches, sensors and actuators interact.

The fluid sensor can be purchased from various online vendors or built yourself if you're feeling saucy. It is a relatively simple circuit that outputs 5V when two metal leads are both in a fluid.

Mechanical:

Because it's often tricky to design a device to produce linear motion from scratch, the cup tray is simply an old CD-ROM drive that was put out to pasture. The pump, which is immersed in the beer container (which can be housed within the BeerBot), is a cheap bilge pump rated at 12V but will drizzle out foamy beer at 5V. Rather than have two power supplies, I decided waiting a few more seconds for beer would only make it taste that much sweeter when the nectar hit my lips.

Possible Improvements:

The possibilities are endless. If your roommates are smarter than mine, you would probably want to increase the number of input combinations (three toggle switches have only eight possible combos). A keypad is the logical solution. Also, the 7-segment LED can be upgraded to an LCD to display more useful information as well as flaunt your electronic savvy.

In my particular project, some of the components are unnecessary (as you can quickly deduce from the wiring diagram). The logic gates, for example, could be eliminated and replaced with slightly creative wiring. But more ICs on your board are useful for better impressing your nerdy friends.

PicBASIC Code:

```
        ' ** define variables ** '
RED    var  PORTA.0                       'RED LED output
CUP    var  PORTA.1                       'CUP     input
CODE   var  PORTA.2                       'CODE    input
ENTER  var  PORTA.3                       'ENTER   input
```

```
SENSOR var  PORTA.4                                     'SENSOR  input
ALARM  var  PORTB.0                                     'ALARM   output
PUMP   var  PORTB.1                             'PUMP    output
RVS    var  PORTB.2                             'output for motor in reverse direction
FWD    var  PORTB.3                             'output for motor in forward direction

pins       var byte[11]                         '11 element array for pin settings
song1      var byte[12]                                 'array for song notes
i          var byte                     'for designating pin
j          var byte                             'counting variable
ms         var byte                     'for alarm counter
tone       var byte                     'tone of buzzer (1 -> 127)
increment  var byte                             'add/subtract increment from tone
alarm_time var byte                             'length of time to sound alarm (in 12 ms)
motor_time var byte                             'length of time to push cup (in ms)
auto_off   var byte                             'max length of time to keep pump on (safety feature)
reset_time var byte                             'max length of time until program is reset (safety feature)
bring_back var byte                             'max length of time to wait, after cup is full, until plate is
                                                'brought back in


      ' ** START ** '
CLEAR                                           'clear registries

      ' ** define port settings ** '
TRISA = %11111110                               'PORTA.0 is LED output, rest are inputs
TRISB = %00000000                               'ALL PORTB pins are outputs

        ' ** define which pins display what ** '
'pins[i] = %ABCD0000                                    'first 4 pins are Qa,Qb,Qc,Qd; last 4 are low
                                                'PORTSB.0,1,2,3 (ALARM,PUMP,RVS,FWD) are off
                                                'PORTSB.4,5,6,7 (Qd,Qc,Qb,Qa) depend on i
pins[0]  = %00000000                                    'diplay 0 (0000)
pins[1]  = %10000000                            'diplay 1 (1000)
pins[2]  = %01000000                                    'diplay 2 (0100)
pins[3]  = %11000000                                    'diplay 3 (1100)
pins[4]  = %00100000                                    'diplay 4 (0010)
pins[5]  = %10100000                                    'diplay 5 (1010)
pins[6]  = %01100000                                    'diplay 6 (0110)
pins[7]  = %11100000                            'diplay 7 (1110)
pins[8]  = %00010000                                    'diplay 8 (0001)
pins[9]  = %10010000                                    'diplay 9 (1001)
pins[10] = %11110000                                    'diplay ? (1111)

        ' ** song array ** '
song1[0] = 65
song1[1] = 69
song1[2] = 73
song1[3] = 77
song1[4] = 82
song1[5] = 87
song1[6] = 92
song1[7] = 97
song1[8] = 103
song1[9] = 110
song1[10] = 116
song1[11] = 123

        ' ** initialize ** '
```

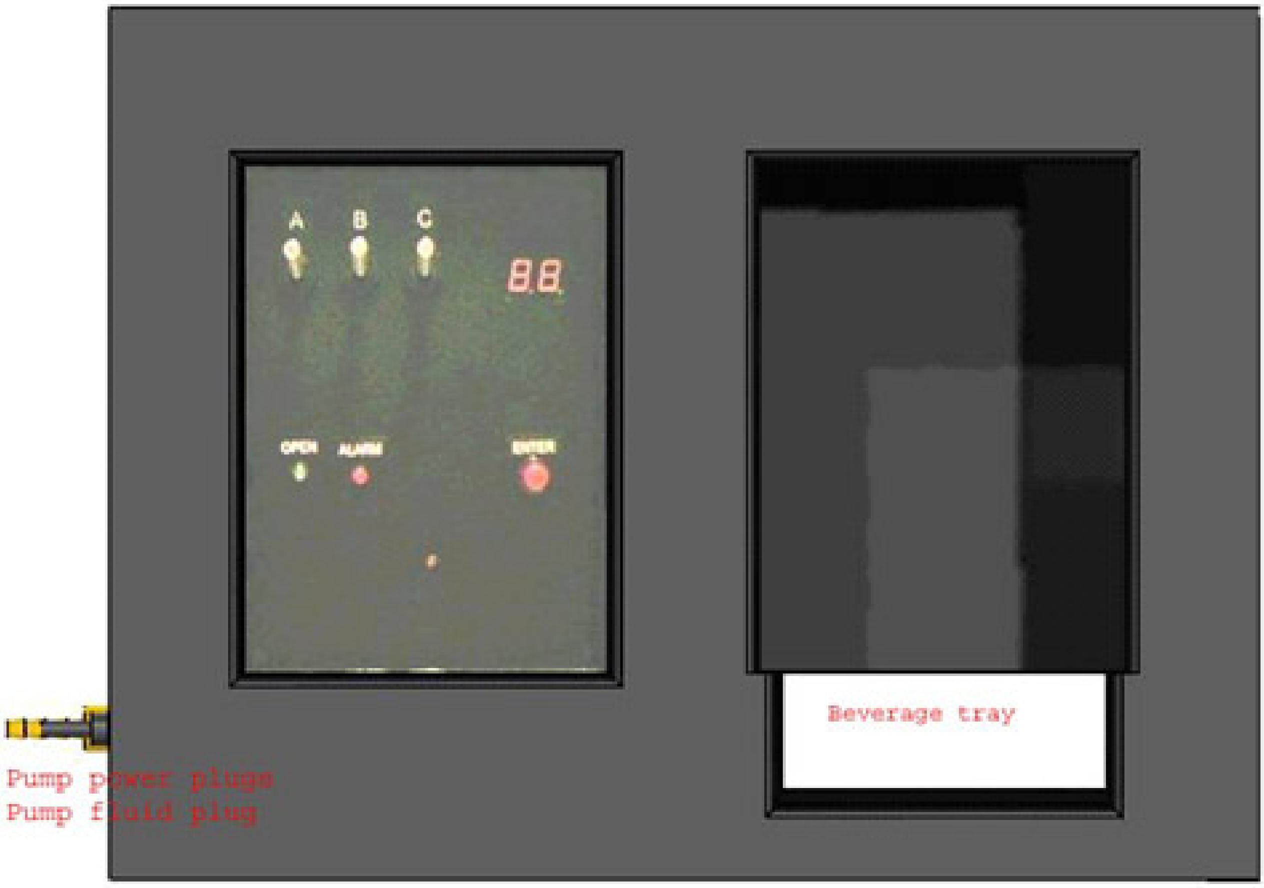
```
motor_time = 75                                 '0.6 seconds (12*.075)
alarm_time = 250                        '3 seconds (12*.250)
auto_off   = 833                        '10 seconds (12*.833)
reset_time = 25000                      '300 seconds (12*2.500)
bring_back = 833                        '10 seconds (12*.833)
increment  = 1
i = 0                                   'reset counter to zero
LOW FWD                                         'make sure motor (in fwd direction) is off
LOW RVS                                         'make sure motor (in rev direction) is off
LOW RED                                         'make sure red LED is off
PORTB = pins[0]                                 'display zero on the 7-seg display
SOUND ALARM,[50,25]                             'beep alarm to signal that power has been turned
on
LOW ALARM                               'turn alarm off
PORTB = pins[10]                        'display nothing on the 7-seg display
PAUSE 50                                'wait 50 ms
PORTB = pins[0]                                 'display zero on the 7-seg display
SOUND ALARM,[50,25]                             'again, signal that power has turned on
LOW ALARM                               'turn alarm off


        ' ** MAIN ** '
MYLOOP:
        LOW RED                                 'turn red LED off
        IF ((CODE=1) AND (ENTER=1)) THEN        'if the code is correct, then
                i = 0                   'reset counter to zero
                PORTB = pins[i]                 'change display
                HIGH FWD                'forward motor
                PAUSE motor_time        'pause for motor_time
                LOW FWD                         'turn off motor
                GOSUB _NOTIFY                   'beep alarm to signal that user needs to do
something
                GOSUB _CUPIN                    'wait for cup to be placed on plate
        ENDIF
        IF ((CODE=0) AND (ENTER=1)) THEN        'if the code is incorrect, then
                i = i+1                 'increment counter
                IF (i>9) THEN           'if counter is at 10, then
                        i = 0           'reset to 0
                ENDIF
                PORTB = pins[i]                 'change display
                HIGH RED                'turn red LED on
                GOSUB _ALARM                    'sound alarm
        ENDIF
        PAUSE 50                        'small debounce
GOTO MYLOOP


        ' ** subroutine for sounding alarm ** '
_ALARM:
        tone = 1                        'start alarm's tone at low frequency
        FOR ms=0 to alarm_time                  'for alarm_time
                IF ((CODE=1) AND (ENTER=1)) THEN        'if correct code is entered, then
                        GOTO MYLOOP             'go back to MYLOOP
                ENDIF
                IF (ENTER=1) THEN       'if ENTER is pressed, then
                        HIGH RED        'turn the red LED on
                ELSE                    'otherwise
                        LOW RED                 'keep the red LED off
                ENDIF
```

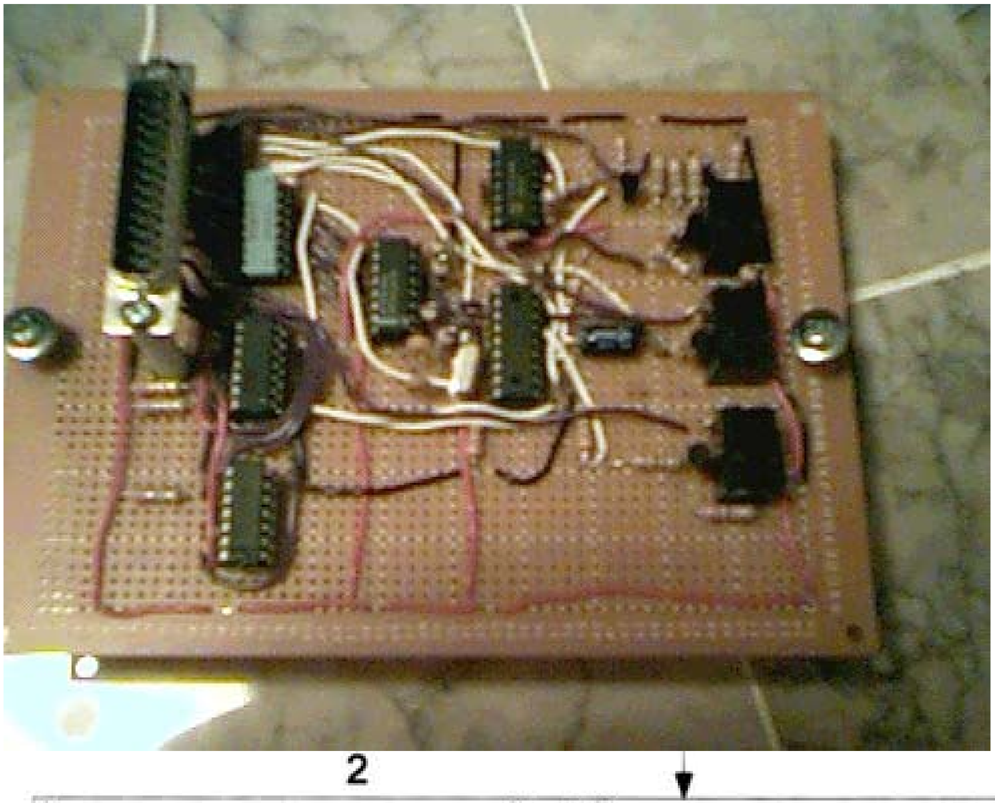
```
      SOUND ALARM,[tone,1]                'sound alarm for 12ms
      tone = tone + increment*2*i         'increment tone
      IF ((tone>127) OR (tone<2)) THEN'if the tone is at its boundaries, then
            increment = -i                'switch the sign (+/-) of increment
      ENDIF
      next ms                             'go back thru the loop
GOTO MYLOOP                               'go back to MYLOOP


      ' ** subroutine for bringing cup in ** '
_CUPIN:
      IF (CUP=1) THEN                     'if the cup button is triggered, then
            PAUSE 20                'wait for 10*.020 seconds and
            IF (CUP=1) THEN               'check it again to make sure a cup is on the plate
                  HIGH RVS          'reverse motor
                  PAUSE motor_time  'for motor_time seconds
                  LOW RVS                 'turn off motor
                  IF (CUP=1) THEN         'if there is still a cup on the plate, then
                        PAUSE 100
                        HIGH PUMP   'turn on the pump
                        GOSUB _FULL'until the cup is full
                  ELSE              'otherwise
                        HIGH FWD    'push the cup back out
                        PAUSE motor_time  'for motor_time seconds
                        LOW FWD           'turn off motor
                  ENDIF
            ENDIF
      ENDIF
GOSUB _CUPIN




      ' ** subroutine for checking to see if the cup is full ** '
_FULL:
      IF (CUP=1) THEN                     'as long as there is a cup on the plate, then
            IF (SENSOR=1) THEN            'if the sensor is tripped, then
                  PAUSE 10          'pause for .01 sec
                  IF (SENSOR=1) THEN      'check it again to make sure, then
                        GOSUB _CUPOUT     'push the cup out
                  ENDIF
            ENDIF
            PAUSE 10                'wait .01 sec
      ELSE                          'if the cup is ever lifted from the plate, then
            GOSUB _CUPOUT                 'push the cup back out
      ENDIF
GOSUB _FULL                         'if the cup hasn't been filled after auto_off time, then
                                    'push it out anyway (to prevent the pump from running
indefinitely)


      ' ** subroutine for pushing the cup back out ** '
_CUPOUT:
      i=0                           'reset counter to zero
      PORTB = pins[i]                     'change display
      LOW PUMP                      'turn off the pump
      PAUSE 500                     'wait for 1/2 second to let pump bilge rest of liquid
      HIGH FWD                      'push the cup out
      PAUSE motor_time              'for motor_time seconds
```

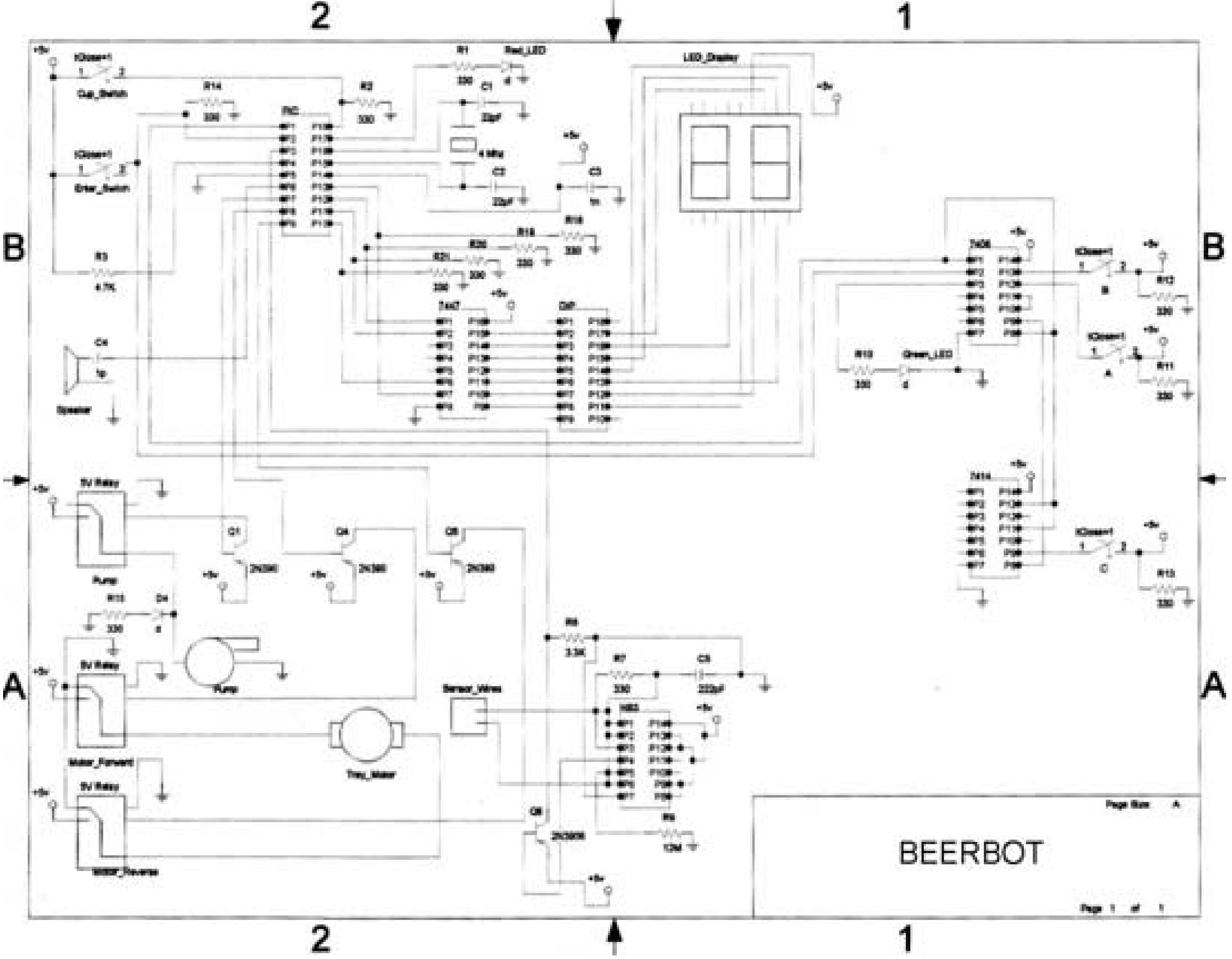
```
     LOW FWD                       'turn off motor
     GOSUB _NOTIFY                 'beep alarm to signal that user needs to do
something
     GOSUB _CUPBACKIN

_CUPBACKIN:
     IF (CUP=0) THEN               'if the cup is removed, then
          HIGH RVS            'reverse motor
          PAUSE motor_time    'for motor_time seconds
          LOW RVS                  'turn off motor
          GOTO MYLOOP              'go back to MYLOOP
     ENDIF
GOSUB _CUPBACKIN              'go back to CUPBACKIN


     ' ** subroutine for beeping alarm to signal that user needs to do something ** '
_NOTIFY:
     FOR ms=0 TO 11
          PORTB = pins[ms]
          SOUND ALARM,[song1[ms],5]
          PAUSE 8
     NEXT ms
          PORTB = pins[i]
     RETURN
```

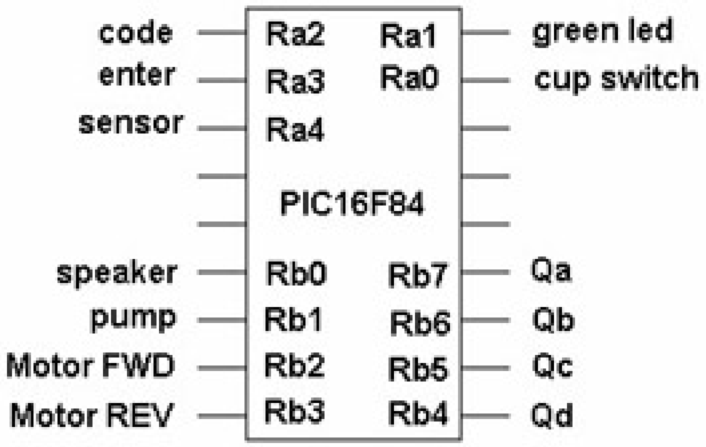
Other graphics:

A
B
C
88
OPEN
ALARM
ENTER


A
B
C
88
OPEN
ALARM
ENTER
Beverage tray
Pump power plugs
Pump fluid plug




2
1
B
A
+5v
IClose=1
Cup_Switch
Enter_Switch
R14
330
R2
330
R1
330
Red_LED
C1
22pF
4 Mhz
C2
22pF
C3
1n
PIC
R3
4.7K
R18
330
R19
330
R20
330
R21
330
7447
DIP
LED_Display
C4
1p
Speaker
7408
7414
R12
330
R11
330
R13
330
R10
330
Green_LED
5V Relay
Pump
R15
330
D4
Q1
Q4
Q5
2N390
Motor_Forward
Motor_Reverse
Tray_Motor
Sensor_Wires
R6
3.3K
R7
330
C5
220pF
NE5
R9
10M
Q6
2N3906
BEERBOT
Page Size: A
Page 1 of 1



See It In Action:

The BeerBot can be viewed in all its glory at
http://www.engr.colostate.edu/~dga/video_demos/mechatronics/index.html#PIC_PROJECTS.

CAD drawings and other information can be found at
http://www.engr.colostate.edu/~ryanf/beerbot.htm