The RoboClock[1]

Fusing Educational Play with Cutting Edge Technology

By

David Deutsch
Manhattan Center for Science and Mathematics
New York, NY 10029

And

MiChelle Carpenter-Smith
Packer Collegiate Institute
Brooklyn, NY 11201

[1] This Work was supported by the National Science Foundation under an RET Site Grant 0227479

ABSTRACT

One of the most challenging skills a young child is asked to master is the telling of time. While there are many digital options for watches and clocks, most educators and parents still believe that the ability to read a traditional timepiece is as valuable as counting money. In order to address the predictable inquiry of "Why do we have to know this?" as well as to bring learning to tell time into the age of microcontrollers and complex circuitry, we have created a prototype for a toy which, through the use of mechatronics and engineering design principles, helps children to learn how to interpret the positions of an hour and minute hand.

TABLE OF CONTENTS

STANDARDS CORRELATION

This project focuses on the integration of mechanical and electrical engineering design principles, computer programming, general physics and mathematics. It supports the following New York State process skills:

Standard 1: To use mathematical analysis, scientific inquiry and engineering design, as appropriate, to pose questions, seek answers and develop solutions;

Standard 6: To understand the relationships and common themes that connect mathematics, science and technology and apply the themes to these and other areas of learning;

Standard 7: To apply the knowledge and thinking skills of mathematics, science and technology to address real-life problems and make informed decisions

CLASSROOM APPLICATION

If this course were to be recreated in the classroom, it could be in the form of discrete laboratory demonstrations and experiments, where students explored the behavior and functions of individual components (such as variable resistors, IR sensors, photoresistors, etc). It could also be recreated as a full semester elective course (Introduction to Engineering, Advanced Computer Topics, etc.) where students were given sufficient theoretical background to design, construct and troubleshoot a project. Implementation depends on several variables: interest of students; math and science ability and background of students; allotted funding; competition with other electives and required courses, expertise of instructor; scheduling conflicts and space constraints; support from parents/administrators.

A full elective course that taught basic engineering and design principles and focused on mechatronics in particular would:

- fully incorporate the above standards;
- engage and challenge students;
- demonstrate interdisciplinary learning;
- give concrete examples of applies math and science;
- create a context for math, science, technology and art teachers to collaborate;
- give artistic students another outlet for creativity;
- expose students to engineering and other technical fields of study
- allow teachers to have more professional challenge
- attract students other courses in math and science

1. Introduction

When we began this course, neither of us had much familiarity with microcontrollers. We each had ideas about interesting ways to use sensors and actuators but had little background about the many types and multiple uses for these devices. We knew how to do basic wiring on a circuit board but had to learn how to use P-basic programming to get individual components to perform to our specifications. Our project is the culmination of four weeks of intensive study of microcontroller and electronic component behavior, as well as a good deal of inventiveness, troubleshooting and "thinking outside of the square". We chose the project for its educational merits, then used this context for demonstrating the functions of several sensors and theories we found intriguing. The result: a prototype of a high tech toy that teaches young children how to tell time with the help of the Basic Stamp 2, and LED display, photo-resistors, a potentiometer, an IR emitter, R-C circuitry and one very feisty robot.

In order to successfully complete this prototype, we needed to have a clear understanding of analog versus digital input and R-C circuitry. We also needed to know how several different types of electronic components functioned and how to read schematics that visually described how these components were electrically connected to one another. We acquired a command of P-basic programming by performing several "pre-experiments" prior to the construction of this prototype.

2. General Theory

2.1 Digital signal input - The Hour hand

The hour hand and minute hand were designed to mimic a "real" clock, by using one shaft imbedded inside another. The hour hand controls the outer shaft, which has a binary encoder attached to it. This encoder is a circle broken into twelve thirty-degree segments. Each segment is divided into four portions. These four portions are shaded black or left clear, to create the binary numbers one through twelve. We programmed the Basic Stamp so that when the photoresistors positioned over it detected light, the number '0' would be displayed on the screen, and a "1' would be displayed when the photoresistors detected darkness. In this way, we created binary code by strategic shading of the portions. Because the hour hand and the encoder are aligned, turning the hour hand turns the encoder so that specific light and dark segments are exposed to the photoresistors. Thus, when the photoresistors "read" the binary code, the Basic Stamp converts this code into a numeral and outputs it to the LED display. Thus we have encoded the position of the hour hand digitally.

Although the angular position of the hour hand is a continuous function, the binary encoder we used to make the hour hand function is discrete. Each segment represents a fixed value and this does not change over the entire segment (or slice). Only when the line that divides segments is crossed is a new value read by the photoresistors. Or so we initially thought. Our experience taught us a lesson known well to those in the encoding world. It is rather unlikely that all four of the light sensors cross the "dividing line" between segments simultaneously. Thus rather than move smoothly from 1 to 2, there is an "in between" state which briefly appears (perhaps a 3) as individual bits shift. Simpson ('87) suggests the "Gray Code" as a solution to this problem. This code involves a method of counting in which only 1 bit changes for successive integers. Refer to the table to compare counting in the decimal system, the binary system, and a variation of the

Gray code. In the next refinement of the RoboClock, we would replace our current binary encoder with the modified Gray encoder.

Decimal	Binary	Modified Gray
1	0001	0011
2	0010	0010
3	0011	0110
4	0100	0111
5	0101	0101
6	0110	0100
7	0111	1100
8	1000	1101
9	1001	1111
10	1010	1110
11	1011	1010
12	1100	1011

2.2.1 Analog signal input – the Minute hand

The inner shaft is connected to a continuous potentiometer (also called a variable resistor). This potentiometer was calibrated through P-basic programming to have sixty segments
(0 - 59) which correspond to the minutes on a clock. When the minute hand is turned, the potentiometer moves from a reference point to a new position. This position is then "read" by the Basic Stamp, converted into a binary number which is then converted into a numeral representing minutes and output on the LED display. This is an example of analog input of information. The potentiometer is an analog device because as it is turned it resistance varies continuously. In theory, this resistance also varies linearly with the angular position of the potentiometer shaft and thus multiplying the potentiometer resistance (or voltage) by some constant should return the position of the minute hand. In practice, we needed to add a small offset to determine the time indicated by the minute hand.

2.2.2 R-C circuitry

How does the Basic Stamp actually determine the resistance of the potentiometer? To understand the answer, one must grasp the functioning of a circuit in which a resistor and a capacitor are connected in series; the "R-C" circuit. The function of a capacitor in a circuit is to store charge, while resistors serve to limit current. If a discharged capacitor in an R-C circuit is suddenly connected to a constant voltage source, it will begin to charge. In the long run, the capacitor will stop all current flow in the circuit, since it will have reached its maximum charge (its "capacity"). The circuit behavior before that time is of interest to physicists. Since resistors limit current, an RC circuit with a large resistance (or capacitance, for that matter) will take a long time to fully charge. Inversely, if the resistance (or capacitance) is low, the capacitor will charge very quickly.

The pattern that emerges is that the product of capacitance and resistance (RC) is the determinant of the time involved in charging the capacitor. One way to state this pattern is that RC represents the time it takes for the capacitor to reach 63 percent of its maximum charge.

The Basic Stamp 2 does not measure charge, but it does input/output (I/O) pins which behave in a very simple manner. For input voltages greater than 1.4 volts, the I/O pins register a "low" input of 0 volts. For voltages between 1.4 volts and 5 volts, a "high" is registered. By connecting an I/O pin of the stamp to an RC circuit as diagrammed below, the input pin effectively measures the voltage across the resistor (in this case, a potentiometer). By discharging the capacitor and then using the P-BASIC command, RCTIME, the Stamp tracks how long it takes for the resistor voltage to fall from 5 volts to 1.4 volts. As the resistance of the potentiometer varies, so does the value of RCTIME in linear fashion. Simple calibration of the minimum and maximum values of RCTIME allows a programmer to scale these values to fit most desired intervals, including one which varies from 0 to 59, as the minute hand of a clock does.

2. Prototype Specifics

The educational toy we created operates in several modes:

I. In this mode, the child-user moves the minute and the hour hands to the desired position then presses a button. The LED display responds by showing the corresponding time (think about a digital watch).

II. In this mode, the P-basic programming tells the Basic Stamp 2 to generate a random time in hours and minutes, and to output this time on the LED display. The child-user must set the hands of the clock face to match the digitally displayed time. If the time is correct to within two minutes, a buzzer will make a congratulatory “happy” sound. If the time is incorrect, a “sad” buzzer sound will result and the child-user will need to try again.

III. This mode is an extension of the second, with the addition of a separately programmed (but integrated) robot. If the clock hands are correctly positioned, the “BOE-bot” will move a certain distance forward. Otherwise, the “BOE-bot” will move a distance backwards. The objective is for the child-user to have enough correctly positioned clock hands so that the “BOE-bot” crosses a pre-set finish line.

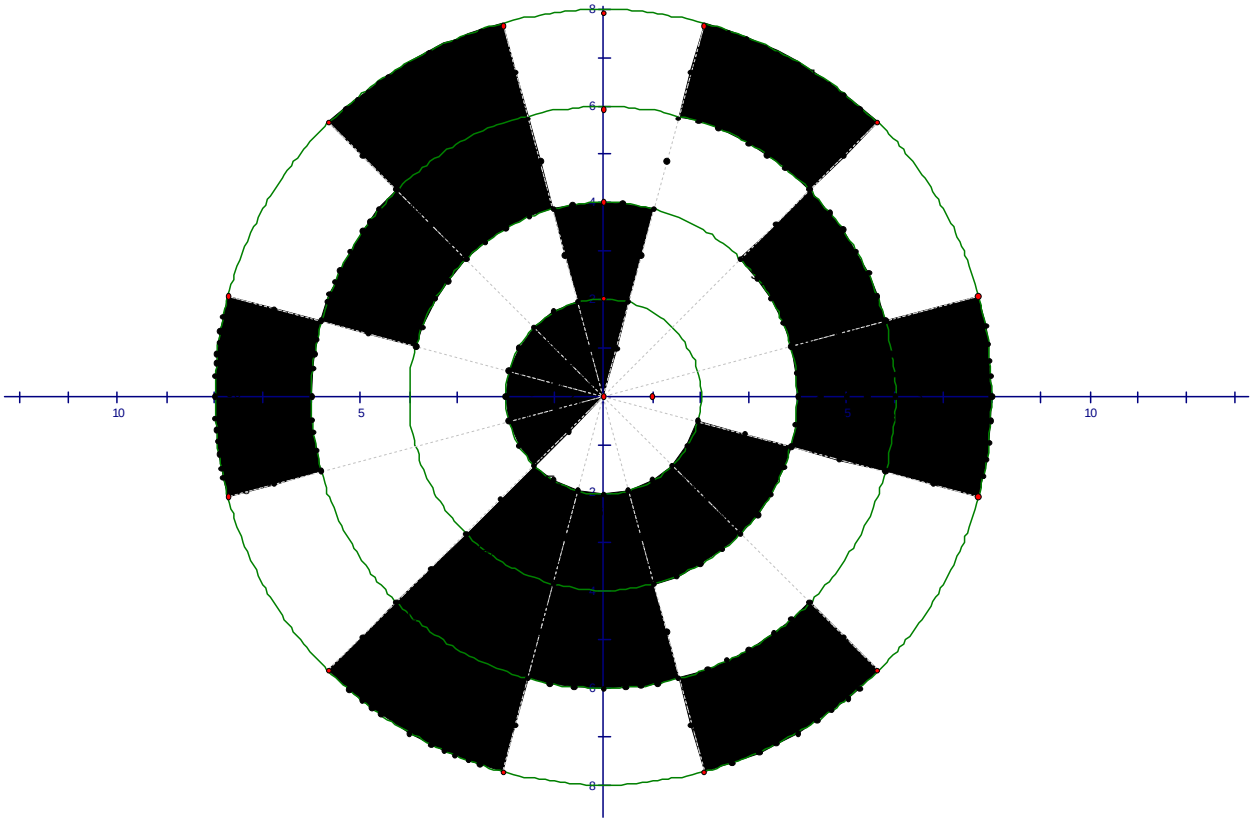


Fig 2.1 The binary encoder

2.1 Equipment List

COMPONENT	WHY WE USED IT
Board of Education (BOE) with Basic Stamp 2	microcontroller & circuit board from Parallax, Inc.
Binary encoder (twelve regions, four levels)	for the photoresistors to "read" light into binary code
4 photoresistors	respond to the presence or absence of light
1 continuous potentiometer	to adjust position of the minute hand
1 multiplexer	to reduce the # of I/O pins needed to run the LED display
1 IR LED	for two-way communication between the clock and the program
1 LED display	to show the time in hours and minutes
1 on-off switch	to control the light bulbs which power the photoresistors
push buttons	to switch from one game-playing mode to another
several 10K ohm resistors	to limit the current flowing from the power source to individual components
1 microfarad capacitor	part of the R-C circuit that controls the calibration of the potentiometer
2 miniature light bulbs with holders	to provide more direct light so photoresistors will be more sensitive
4 AA batteries with holder	power source for the light bulbs
standard parts for the parallax ro-bot	these include 2 servo-motors, another BS2, IR sensors/emitters, etc
wires of various lengths	for making complete circuits between components
1 15-pin data port	to transfer the P-basic program from the computer to the microcontroller
1 speaker	to make a programmed "win" or "lose" sound
1 IR emitter	for two-way communication between the clock and the program

3. Design analysis and troubleshooting

If the child user sets the clock to 3:00, the LED will output the corresponding time, but if the clock is set to 2:58 the way a real clock would actually display it, the LED display will incorrectly read "3:58". As the minute hand of a real clock moves, the hour hand moves continuously with it. Therefore by 2:58, the hour hand is exceedingly close to 3. For our encoder, when the hour hand points to 3, the sensors are actually above the middle of the "slice" that encodes a 3, not on the border between a 2 and 3. We might partly solve this problem by rotating the encoder about 20 degrees so that "3" is no longer in the center of its segment. This method does not completely solve the problem because it increases the possibility of an "in between" state mentioned earlier and which could be solved by using a modified Gray code. Alternatively, future versions of the Roboclock might altogether skip the digital encoding and simply use a potentiometer for the hour hand. It took many hours of experimentation and troubleshooting to properly calibrate the binary encoder, whereas the potentiometer was a fairly simple component to incorporate. We considered replacing the encoder with another potentiometer, but this would have denied us the opportunity to create an example of digital input. Given that so much of current technology uses digital input (think compact disc), we thought that incorporating an example of it was especially instructional. One might also note that a child who knows to place the hour hand exactly between the 4 and the 5 at 4:30 has mastered the art of telling time and may no longer need the Roboclock.

On a real clock, there is no sixtieth minute, only 0 through 59 before the hour hand increments by one. Occasionally, our LED will incorrectly read 0-60 as the minute hand is making a rotation. This is because the potentiometer has a small gap (it does not make a completely closed loop) that affects the accuracy of our calibrations. We solved this problem through the programming code: if the minute hand was about to display "60" it was instructed to simply replace the "60" with "00".

We initially designed the photoresistors to be activated by four light bulbs that were positioned equidistantly from the center shaft. We discovered that this placement was not ideal because the photoresistors needed a more direct source of light. We solved this problem by removing two of the light bulbs and placing the remaining two directly under the armature that holds the photoresistors.

The armature for the photoresistors was its own prototype, first made out of a thin piece of stainless steel (which compromised the function of the photoresistors because of metal-to-metal contact), then two different thicknesses of plexi-glass (which didn't work because the plexi-glass was too thin to drill through and still remain strong). We finally chose opaque high-strength nylon. Another problem was that the leads to the photoresistors broke several times during the process of handling (we needed to adjust their closeness to the light bulbs by pulling them through the holes in the armature). We recommend that in the next iteration, the photoresistors should be imbedded in a section of breadboard and permanently wired.

We designed the box encasing the toy's components to be rather large so that we could have easy access for repairs and adjustments. Further versions would be designed to greatly reduce the size to something more like that of a child's toy.

Due to lack of time, we were unable to create a track (or more ideally, a maze) that the robot would move along as the child-user logged his or her responses. Also, the IR sensors on the robot were not sensitive enough for it to respond to a right/wrong answer if it was more than several feet from the IR emitter on the breadboard. We did not have time to address this issue, so we tested the robot by setting it on a box near the toy and noting whether it functioned within this limited range.

A further improvement involves incorporating a voice chip that speaks the time as it was displayed on the LED. Other levels of detailed programming include a voice chip that would say, "The hour hand is correct but the minute hand is not". Time constraints prevented this from happening.

We believe that this project challenged us tremendously. We encountered mechanical difficulties in the workshop, as well as programming and circuit challenges in the lab. We are satisfied with the outcome - it agrees more than not with our initial proposal. The authors also feel more mature professionally. This project simultaneously enthused us to the excitement and true learning that a long term project can inspire in a student while reminding to beware the humility and frustration of the student who doesn't immediately grasp a concept.

4. Marketing Proposal

We believe that this educational toy will have consumer merit with upwardly mobile parents who want their children to begin learning at an early age. We also believe that it will be a trailblazer in a market that is becoming disenchanted with traditional computer games yet ever more interested in computerized interaction.

We propose that this product be piloted to teachers in early grade classrooms and to the parents of their students. We also envision several other types of instructional play:

- Changing the clock face numbers from Arabic to Roman, Binary and Hexadecimal to teach and challenge the mathematically gifted child;
- Using an alpha-numeric LED display to teach young children the alphabet and how to spell simple words;
- Replacing the clock face with other “boards” designed to drill basic addition and multiplication;

This type of product would be available in stores such as The Sharper Image, Zany Brainy, Einstein’s World and others venues that sold educational toys and/or software products. Given the cost of inexpensive, limited memory microcontrollers, the cost for the toy including several interchangeable boards, the robot and a game track could be about $20. For the hobbyist parent who is interested in microcontroller functions, we would also market a version where the robot (and its Board of Education) would need to be wired, assembled and programmed. We are hopeful that Parallax, Inc. would be interested in a partnership with Polytechnic University and the authors to patent and market a refined version of this product.

5. Acknowledgments

We would like to thankfully acknowledge the visionary leadership of Professor Vikram Kapila, director of the Mechatronics Laboratory at Polytechnic University. Also, we are most appreciative of the many hours of instruction, troubleshooting and cheerleading from Project Instructor Sang-Hoon (Nathan) Lee, and teaching assistants Yan Fang (Yvonne) Li, and Hong Wong. We must acknowledge that without the expertise, patience and creativity of Alessandro Betti, we would have nothing to show for our efforts. Finally, we are thankful to our ever-helpful graduate students, to Parallax, Inc. for its generous donation of materials and to Polytechnic University and the National Science Foundation. By making such cutting-edge work possible to bring into the classroom, students who otherwise might never know that such things existed, will now have a sense about what is possible.

Appendix

```
'{$STAMP BS2}
'{$PBASIC 2.5}

'ClockNow.bs
'Michelle Carpenter-Smith and David Deutsch
'July 30, 2003

'************ I/O Definitions *******
DataPin     CON 0                   'Pins to communicate with the
Load        CON 1                   'MAX 7219 -- Drives the LED display
Clock       CON 2
Speaker     CON 3                   'Indicate right or wrong answer
IR_Send     CON 4                   'Communicate with the robot (move it!!)
PotPin      CON 9                   'pot wiper connects to pin 9 (minute hand)
Switch1     CON 10                  'To display time digitally (Mode 0) or
                                    'Display a random time (Mode 1)
Switch2     CON 12                  'Correlates clock hands to LED (Mode 0)
                                    'Generates random time, tests answer (Mode1)

Blank       CON %1111
Decode      CON $09                 'bcd decode register
Brite       CON $0A                 'intensity register
Scan        CON $0B                 'scan limit register
ShutDn      CON $0C                 'shutdown register (1 = on)
Test        CON $0F                 'display test mode

'G major scale:

G       CON 3136                    'G note
A       CON 3519                    'A note
B       CON 3950                    'B note
C       CON 4182                    'C note
D       CON 4696                    'D note
E       CON 5272                    'E note
Fsharp  CON 5914                    'Fsharp

'************ Variables *************
d7219       VAR Byte                'Send data serially to display driver
index       VAR Byte
tone        VAR Word                'To be used with freqout
direct      VAR Byte                'Tells the robot to move back or forward
dur         VAR Word                'Time duration of a single note
beat        CON 100                 'single beat
rawValue    VAR Word                'raw value from potentiometer in RCTIME
```

```
minute      VAR Word            'scaled value for minute
hour        VAR Byte            '4 bit value determined by encoder/photoresistors
time        VAR Word            'merge minute and hour data
digit       VAR Nib             'Select the LED to light via multiplexer
Mode        VAR Bit             'Correlate time or play a game?
Locate      VAR Byte            'Keep track of robot position
TargetMinute VAR Byte           'Randomly generated minute
TargetTime   VAR Word           'Combined randomly generated minute and hour
MinOffset    VAR Nib            'Necessary in calibrating the minute hand



Initialize:
 DEBUG HOME, ? Mode
 DIRA = %1111                   'Pins 0-3 are all outputs
 Locate = 2                     'Robot starts at position 2


 FOR index = 0 TO 7             'Initializes the MAX7219
  LOOKUP index, [Scan, 3, Brite, 15, Decode, $0F, ShutDn, 1], d7219
  SHIFTOUT DataPin, Clock, MSBFIRST, [d7219]
  PULSOUT Load, 5
 NEXT
 GOSUB ClearClock               'Otherwise it starts with a random display!



Main:

 BRANCH Mode, [Mode_0, Mode_1]

 Mode_0:                        'In this mode, the user moves the clock
  GOSUB CheckMode               'hands.  When he or she presses button2, the
  'DEBUG HOME, ? IN12           'LED display shows the time
  IF IN12 = 1 THEN GOTO Main
  PAUSE 200
  GOSUB ReadTime
  GOSUB WriteTime
  GOTO Main

 Mode_1:                        'In this mode the user plays a game.  Pressing
  GOSUB CheckMode               'button 2 generates a random time and displays it
  RANDOM TargetMinute           'on the LEDs.  The user must then move the hands
  RANDOM TargetTime             'to within 3 minutes of that time.  Pressing button
  IF IN12 = 1 THEN GOTO Mode_1  '2 again will indicate that the computer should
check
```

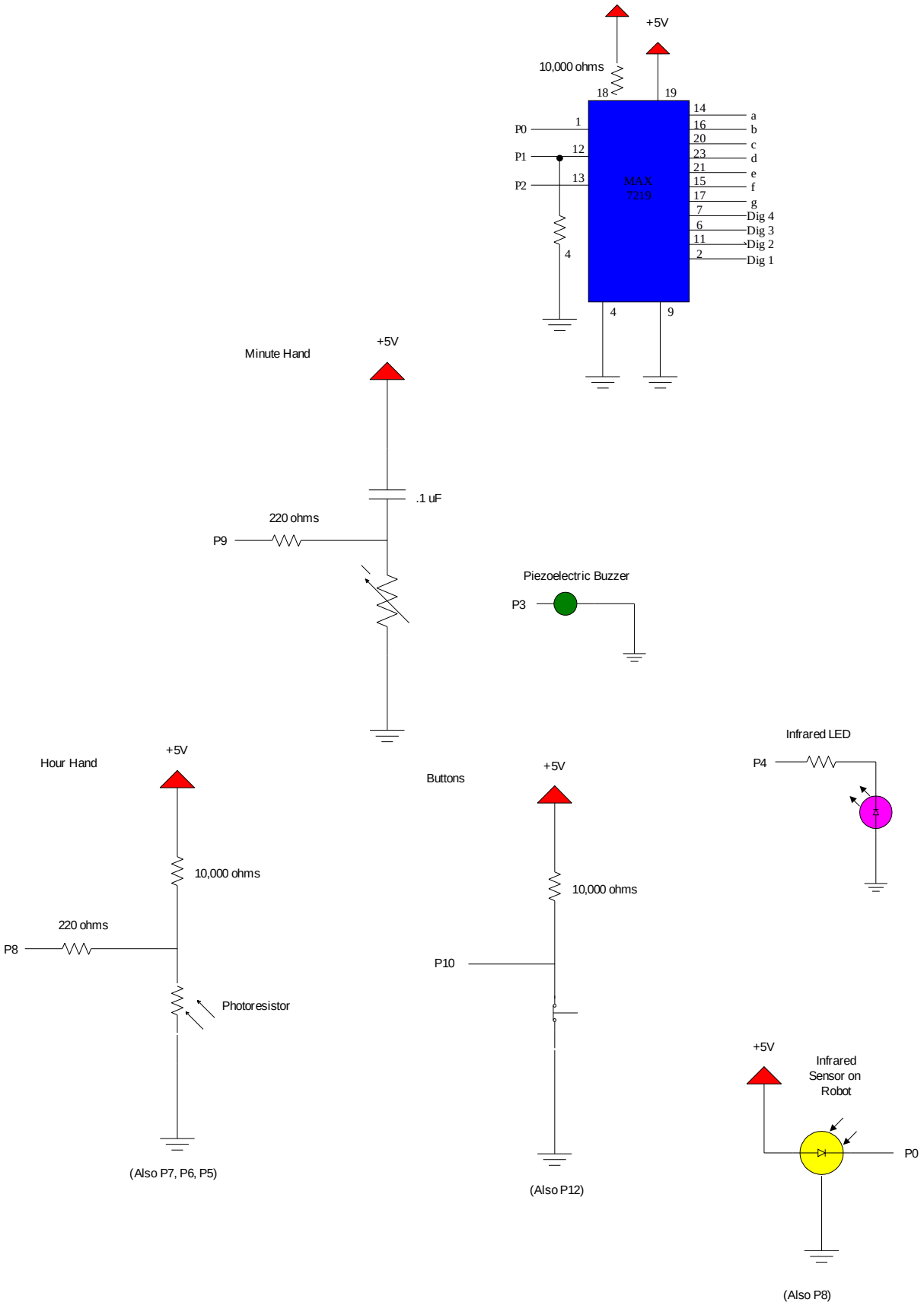
```
                                'the answer.  If correct, a "happy" buzz
place:                          'occurs and the robot steps forward one step.  If
  IF IN12 = 0 THEN GOTO Place       'wrong, a "sad" buzz precedes a backward step
TargetMinute = TargetMinute//60
Time = (TargetTime//12 + 1)*100 + TargetMinute
TargetTime = Time
DEBUG ? Time, CR
GOSUB WriteTime
Hold:
  GOSUB CheckMode
  IF IN12 = 1 THEN GOTO Hold
Place2:
  IF IN12 = 0 THEN GOTO Place2
GOSUB ReadTime
IF ABS(Time - TargetTime) <= 3 THEN GOTO Happy      'User is correct!

Sad:                                    'Or not!
  Locate = Locate - 1
  FREQOUT Speaker, 1000, 100                    'Sad Buzz!
  DEBUG ? TargetTime
  DEBUG? Time
  DEBUG "Try again!!!", CR
  direct = 100                          'This step distinguishes forward from backward
  GOSUB MoveRobot
  IF Locate = 0 THEN GOSUB EndSadly
  GOTO Hold

Happy:
  Locate = Locate + 1
  DEBUG "You Win!!!",CR
  FOR index = 0 TO 3                            'Happy Buzz!
    LOOKUP index, [665, 795, 995, 1320], tone
    FREQOUT Speaker, 100, tone
  NEXT
  direct = 200                          'Will cause forward motion
  GOSUB MoveRobot
  IF Locate = 4 THEN GOSUB EndHappily
  GOTO Main



'************ Subroutines ************

CheckMode:
 IF IN10 = 1 THEN RETURN                        'If button 1 is pushed, change modes.
```

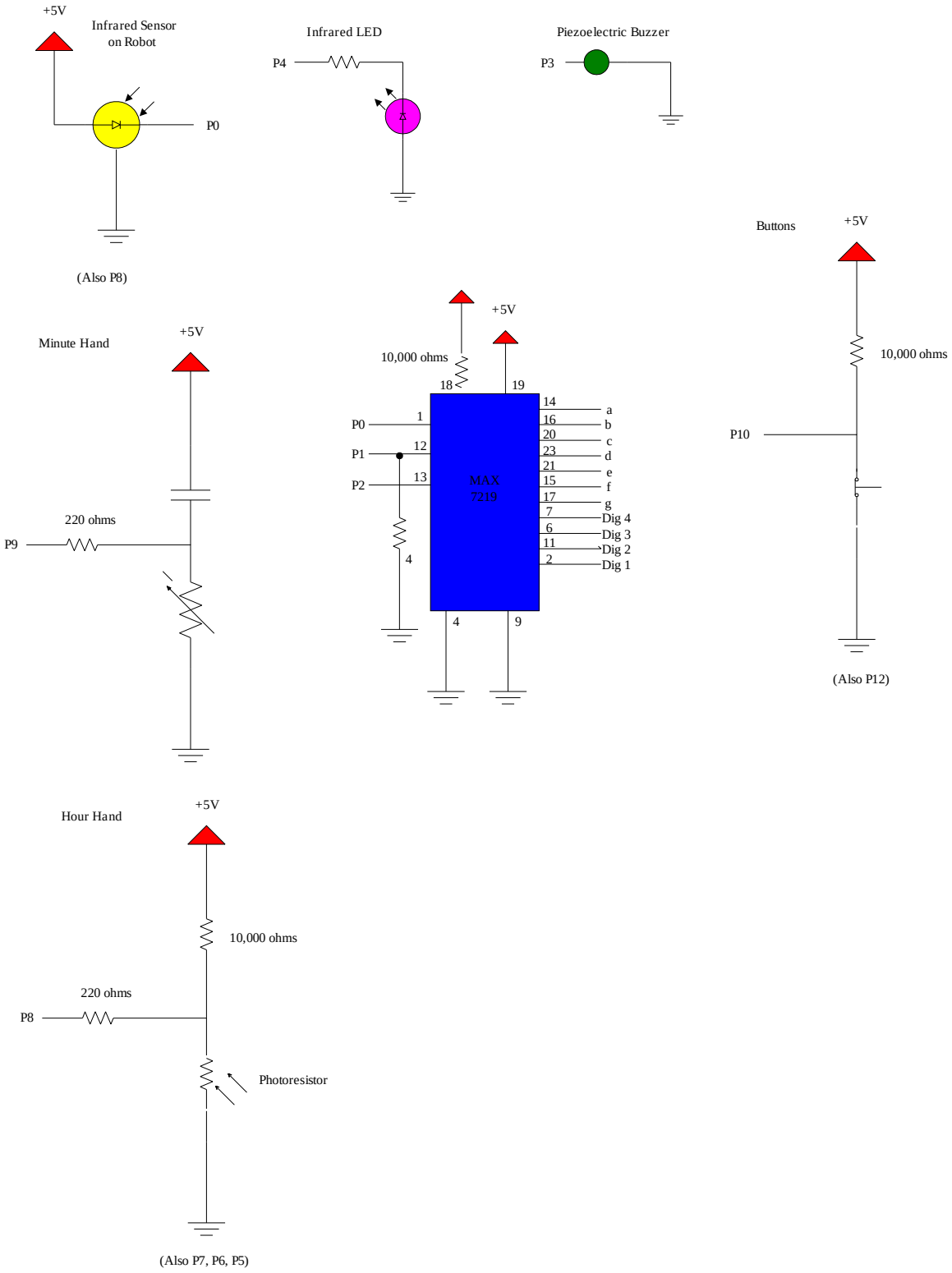
```
ChangeMode:
IF IN10 = 0 THEN GOTO ChangeMode
Mode = Mode + 1
DEBUG CLS, ? Mode, CR
BRANCH Mode, [Mode_0, Mode_1]
RETURN

ReadTime:
HIGH PotPin                                  'Discharge capacitor
PAUSE 1                                      'for 1 millisecond
RCTIME PotPin, 1, rawValue                   'read the pot
MinOffset = 0
IF (rawValue > 2) AND (rawValue < 330) THEN MinOffset = 2 'Offsetting determined
by calibration
IF rawValue > 330 THEN MinOffset = 1
minute = (60*rawValue)/683 + MinOffset
minute = minute//60                          'Avoid occasional trouble with 61st minute!
'DEBUG CLS, ? rawValue, ? Minute

hour.BIT0 = IN8                              'Here the photoresistor value is
hour.BIT1 = IN7                              'determined by the encoder
hour.BIT2 = IN6
hour.BIT3 = IN5
'DEBUG HOME,? hour.BIT0
'DEBUG ? hour.BIT1
'DEBUG ? hour.BIT2
'DEBUG ? hour.BIT3
'DEBUG DEC2 hour

time = 100*hour + minute                     'time is a 4 digit number
RETURN

WriteTime:
GOSUB ClearClock
FOR index = 1 TO 4                           'Instruct the MAX7219 to serially
  d7219 = time DIG (index-1)                 'send data to each of the 4 LED
  IF (index = 4) AND (time < 1000) THEN d7219 = blank  'displays
  SHIFTOUT DataPin, Clock, MSBFIRST, [index, d7219]
  PULSOUT Load, 5
NEXT
RETURN

ClearClock:                                  'If the clock is not cleared at
d7219 = blank                                'each pass, "ghosting" of the
FOR index = 4 TO 1                           'display occurs
  SHIFTOUT DataPin, Clock, MSBFIRST, [index, d7219]
```

```
          PULSOUT Load, 5
        NEXT
        RETURN

       MoveRobot:                          'Send a simple signal to the
        DEBUG "MoveRobot!!!", CR                    'robot.  The time duration for moving
        FREQOUT IR_Send, direct, 38500              'forward is twice as long as for
backward
        RETURN

       EndSadly:
        PAUSE 1000
        direct = 200                       'Send the robot forward
        GOSUB MoveRobot
        PAUSE 1000
        GOSUB MoveRobot                           'To it's original position without fanfare
        Locate = 2                         'Robot back in "home" position
        Mode = 0                           'Change to practice mode
        GOTO main                            'Start over
        RETURN


       EndHappily:
         DEBUG "EndHappily!"
         PAUSE 3000
        joy:                               'Plays the "Ode to Joy"
         FOR index = 0 TO 29
           LOOKUP index,
[B,B,C,D,D,C,B,A,G,G,A,B,B,A,A,B,B,C,D,D,C,B,A,G,G,A,B,A,G,G],tone
           LOOKUP index, [2,2,2,2,2,2,2,2,2,2,2,2,3,1,4,2,2,2,2,2,2,2,2,2,2,2,3,1,4], dur
           dur = dur * beat
           FREQOUT 3, dur, tone
         NEXT
        Locate = 2
 RETURN
```

```
'{$STAMP BS2}
'{$PBASIC 2.5}

'RoboClock.bs
'Michelle Carpenter-Smith and David Deutsch
'July 30, 2003

'************ RoboClock! **********************

'************ I/O Declarations ***************


'************ Variable Declarations **********
counter       VAR Word              'Used to time the length of incoming IR signal
LeftIRDet     VAR Bit               'Stores L IR detection
RightIRDet    VAR Bit               'Stores R IR detection
Index         VAR Byte              'For looping
TwoStep       VAR Nib               'Also for looping
Locate        VAR Byte              'Keep track of robot location
Speaker       CON 2

'----- Initialization ------------
LOW 13                          ' Initialize servo line startup values.
LOW 12
counter = 0
Locate = 2                      'Position 2 at middle of track

OUTPUT 2
FREQOUT 2,500,3000                  ' Signal program is starting/restarting.

DEBUG CLS


Main:
 counter = 0                    'Reset IR signal timer
 RightIRDet = IN0                   'Is the robot being signalled by the clock?
 LeftIRDet = IN8
 DEBUG HOME,? RightIRDet, CR
 DEBUG ? LeftIRDet, CR
 IF (RightIRDet = 0)OR (LeftIRDet= 0) THEN GOTO DetectMode  'Signal is detected if
true
 GOTO main                      'Wait for a signal

 DetectMode:
   RightIRDet = IN0
   LeftIRDet = IN8
```

```
        counter = counter + 1                       'Advance the timer...
        IF (RightIRDet = 0)OR (LeftIRDet = 0)THEN GOTO DetectMode '...until a signal is
no longer detected
        'DEBUG ? counter
        IF counter < 75 THEN GOSUB Step_Backward        'If greater than 75, it will
move forward
        Locate = Locate + 1
        FOR index = 1 TO 50                     'Advances the robot 1 step
          PULSOUT 12, 500
          PULSOUT 13, 1000
          PAUSE 20
        NEXT
        IF Locate = 4 THEN GOSUB EndHappily             '4 is the finish line!
        GOTO Main

        Step_Backward:
        Locate = Locate - 1
        FOR index = 1 TO 50                     'Robot steps once backward
          PULSOUT 13, 500
          PULSOUT 12, 1000
          PAUSE 20
        NEXT
        IF Locate = 0 THEN GOSUB EndSadly               '0 is the other finish line!
        GOTO Main


      '********************* Subroutines ************************


      EndHappily:

        FOR TwoStep = 1 TO 2
          FOR index = 1 TO 50                   'Robot steps twice backward
            PULSOUT 13, 500
            PULSOUT 12, 1000
            PAUSE 20
          NEXT
        NEXT
        Locate = 2                          'Robot will return to "Home"
        RETURN

      EndSadly:
        FREQOUT Speaker, 2000, 100                  'Sad Buzzer!
        Locate = 2                          'Robot will return to "Home"
RETURN                                  '(Motion completed by ClockNow)
```


+5V
10,000 ohms
18
19
P0
1
P1
12
P2
13
4
MAX
7219
14
16
20
23
21
15
17
7
6
11
2
a
b
c
d
e
f
g
Dig 4
Dig 3
Dig 2
Dig 1
4
9
Minute Hand
+5V
.1 uF
220 ohms
P9
Piezoelectric Buzzer
P3
Hour Hand
+5V
10,000 ohms
220 ohms
P8
Photoresistor
(Also P7, P6, P5)
Buttons
+5V
10,000 ohms
P10
(Also P12)
Infrared LED
P4
+5V
Infrared
Sensor on
Robot
P0
(Also P8)


+5V
Infrared Sensor on Robot
P0
(Also P8)
Infrared LED
P4
Piezoelectric Buzzer
P3
Buttons
+5V
10,000 ohms
P10
(Also P12)
Minute Hand
+5V
220 ohms
P9
+5V
10,000 ohms
18
19
P0
1
P1
12
P2
13
4
MAX 7219
14
a
16
b
20
c
23
d
21
e
15
f
17
g
7
Dig 4
6
Dig 3
11
Dig 2
2
Dig 1
4
9
Hour Hand
+5V
10,000 ohms
220 ohms
P8
Photoresistor
(Also P7, P6, P5)

7. References

[1] Online: http://www.parallax.com, web site of Parallax, Inc. manufacturer and distributor of Basic Stamp microcontrollers containing technical specifications and user manuals.

[2] Online: http://www.radioshack.com, web site of RadioShack, a retailer of electrical components.

[3] Online:http://search.nap.edu/readingroom/books/nses/html/overview.html#program, National Science standards

[4] Online: http://standards.nctm.org/, National Mathematics Standards

[5] Online: http://mechatronics.poly.edu/smart/