Final Project

Final Projects will be graded during the last lab meeting of the semester. Projects are intended to be completed by individuals.

The lab instructor will assign a project to you; however,

- you may *request* a project from those listed below, or
- you may make a *written suggestion* describing an alternative project. Suggestions for projects other than those listed below must be as detailed and explicit as those below before they will be considered.

The lab instructor will determine whether your request or suggestion is appropriate. Although requests are usually honored, there is no guarantee that the lab instructor will grant your request or suggestion. Reasons for not honoring your request may include (but are not limited to) too many people wanting to do the same project, and a project having a degree of difficulty that is not appropriate for you personally. Furthermore, your lab instructor may add or subtract requirements from any project. It is the responsibility of the student to resolve possible interpretations of the following descriptions *before* the projects are graded.

Rules applying to all projects

All projects must start up in a well-defined state when the project box is turned on. Unless otherwise indicated, no project should require the project box ON switch to be toggled as part of its subsequent operation.

Your grade will be partially based on neatness. This includes using wires of the appropriate lengths and colors. Although you may assign colors of your choice to various other tasks or concepts, all V_{cc} wires must be red, and all ground wires must be black.

Suggested Final Projects

In the following descriptions, switches will be designated as momentary (i.e., spring loaded like the set of two in the PAD trainers), or double-throw (i.e., can remain in either of two positions, like the set of eight in the PAD trainers). A double-throw switch may be used in place of a momentary switch; when using the circuit, we'll just make sure to always throw that switch down and up, with no intervening action. Note that the double-throw switches in the PAD trainers are *not* debounced.

Projects have different *estimated* difficulty ratings. In grading, projects with lower difficulty ratings will be held to higher standards of neatness, ease of use, clarity of function, etc.

Projects using the BASIC stamp are allowed, but must also include a significant TTL component. How the function is divided between stamp and TTL circuitry must be explicit from the beginning.

- Some suggestions for projects using a stamp are at the end of this list; a request for one of those projects must indicate which portions would be on the stamp.
- A stamp might be used to complete one of the non-stamp suggestions. This will typically lower its difficulty rating significantly.
- Any other proposal for a project using a stamp will be handled as described above for new suggestion. The written suggestion must explicitly outline the part of the project to be completed with TTL.

Project Descriptions Page

Projects Designed & Rated without a BASIC Stamp

Projects Designed & Rated including a BASIC Stamp

Generating "Random" Numbers

Projects Designed & Rated without a BASIC Stamp

1. Traffic Light (Difficulty: 8)

A traffic light is at the intersection of a relatively large highway and a small residential street. Generally, the light is green for vehicles traveling along the highway, and red for the side street. When a car needs to enter the highway from the side street, a sensor in the road triggers (treat as a pulse of unknown width). Generally, when the sensor is tripped, the lights will both change. However:

a. Before changing to red, a light must be yellow for ~2 seconds.
b. A light will become green within 2 seconds of when the opposite light changes to red, but not before it changes to red.
c. When the side street light becomes green, it must *stay* green for 20 seconds. When the highway light becomes green, it must *stay* green for a minimum of 20 seconds.
d. The sensor may be triggered or retriggered at any time during the red-green-yellow cycle. If a new trigger occurs while the side street has a green light, there is no effect. If a trigger occurs while the side street is red, the highway will go immediately to yellow (except that rule c above may delay it). If a car re-triggers the detector while the side street is "yellow", assume that it will not enter the highway. So, in this circumstance, the side street yellow will finish normally, then the main highway will be green for ~20 seconds, then the highway will be yellow for ~2 seconds, and then the main highway will turn red while the side street turns green.

Input summary: 1 momentary switch (road sensor)
Output summary: 6 LEDs (2 traffic lights; note: we have colored LEDs)

2. Railroad Crossing (Difficulty: 6)

A traffic light is located at a railroad crossing. The track is used by both eastbound and westbound trains. There are three sensors along the railroad track: one at the crossing, and one on either side of it, ½ mile from the crossing. At all times, each sensor indicates whether some portion of a train is at that location. Trains may be between ¼ mile and 1¼ mile long, so that at any moment 0, 1, 2, or 3 of the sensors may be active. Note that there can even be a short train on that stretch of track, but with no sensors active at all. Thanks to the good work of the railroad, there is never more than one train on this section of track.

a. Normally, the traffic light is green. However, when a train approaches within ½ mile of the crossing, the traffic light must go yellow for ~2 seconds, and then stay red until the train is completely past the crossing.
b. When the train is completely past the crossing the light returns directly to green (even though part or a train may still be on this stretch of track).
c. Trains do not reverse direction while in the vicinity of the crossing.

Input summary: 3 double-throw switches (the sensors at the crossing and at ± ½ mile)
Output summary: 3 LEDs (traffic light; note: we have colored LEDs)

2b. Enhanced Railroad Crossing (Difficulty: 7)

There is also a sensor in the road at the crossing, which at all times indicates whether there is something there (either a car or a train will activate sensor). On either side of the crossing, ¼ mile from it along the track, there are two warning lights that warn trains if there is a car on the crossing. These light only if there is known to be a train approaching from that direction, and you may assume

that a car will not cross the tracks while the crossing light is red; thus, these only activate if a car is on the crossing while the light is yellow.

Additional Input summary: 1 double-throw switch (the additional sensor at the crossing)
[This switch must be immediately next to the train-at-crossing switch, so that they can be thrown simultaneously when the train crosses.]

Additional Output summary: 2 LEDs (warning lights)

3. Vending Machine (Difficulty: 8)

A vending machine has two selections: pretzels (20¢), and snack cakes (40¢). *First*, a selection is made. *Then*, coins are entered into the machine (nickels, dimes, and quarters only). When the amount of coins is sufficient, the item is dispensed and change is returned.

a. A LED will indicate when a successful sale is made (one LED for each product). This LED should light for ~1 second before the vending machine resets itself for future purchases.
b. After a sale, use a 7-segment display to indicate the number of *nickels* returned as change. This value should be displayed for ~1 second.
c. The machine should have a "coin return" switch that clears the selection, returns all coins as change (see b above), and resets the machine.
d. The vending machine initially contains only 9 of each item. When an item becomes unavailable, an "empty" light comes on. If the user attempts to purchase such an item, a "make another selection" LED lights for ~ 1 second. You may implement resetting the counters by turning the power off and on.

Input summary: 6 momentary switches (3 coin sensors, 2 product selectors, and 1 coin return)

Output summary: 5 LEDs (sale of pretzels, sale of snack cakes, out of pretzels, out of snack cakes, and make another selection)
1 7-segment display (change)

4. Advanced Vending Machine (Difficulty: 10)

A vending machine has two selections: pretzels (20¢), and snack cakes (40¢). Coins are entered into the machine (nickels, dimes, and quarters only) first. When a selection is made, if funds are sufficient, the item is dispensed and change is returned.

a. A LED will indicate when a successful sale is made (one LED for each product). This LED should light for ~1 second.
b. If a selection is made, but funds are insufficient, an LED should light for ~1 second. During this time, the selections should be locked out. During this time, you may assume that the user will not input new coins.
c. After a sale, use a 7-segment display to indicate the number of *nickels* returned as change. This value should be displayed for ~1 second before the vending machine resets itself for future purchases.
d. Once more than 40¢ is input by a user, the machine does not have to accept more coins. Turn on a LED to indicate when the machine will not accept more money. When this LED is on, we'll assume that new coins are immediately returned (with no indication on the 7-seg display).
e. The machine should have a "coin return" switch, which returns all change and resets the machine. All coins returned should be displayed as change as in (c) above.

Input summary: 6 momentary switches (3 coin sensors, 2 product selectors, and 1 coin return)

Output summary: 4 LEDs (sale of pretzels, sale of snack cakes, insufficient funds, and maximum funds)
1 7-segment display (change)

4b. Enhanced Advanced Vending Machine (Difficulty: 11)

The vending machine initially contains only 9 of each item. When an item becomes unavailable, an "empty" light comes on. If the user attempts to purchase such an item, a "make another selection" LED lights for ~ 1 second. You may implement resetting the counters by turning the power off and on.

Additional Output summary: 3 LEDs (out of pretzels, out of snack cakes, make another selection)

5. Automatic Geiger Counter (Difficulty: 6)

An Automatic Geiger Counter (AGC) has a two-digit 7-segment display showing "counts", which indicate the number of radiation units detected over a 4 second period. The AGC has the following operation:

a. A new radioactive sample enters the detection chamber every 8 seconds.
b. For the first 4 seconds, the AGC counts the number of radiation "hits". The main clock on the project kit will be used to create a pulse train to represent this sample; your instructor will adjust the frequency knob to higher values to simulate a more radioactive sample. As a result, you will not be able to use the main clock for timing purposes. Instead, you will have to use a 555 chip. During these 4 seconds, the counter display must be blank.
c. The final radiation count for that sample (from b above) is held and displayed during the final 4 seconds.
d. There must be a manual over-ride momentary switch, which can only activate when the AGC is in the "hold and display" mode. When this switch is toggled, the "hold and display" mode will remain on indefinitely. Normal operation (beginning at (a) above) will resume when this switch is toggled a second time.
e. If the number of counts measured (during (b) above) is greater than 99, then a warning LED will light (during the display phase). When this LED lights, it should have exactly the same effect as toggling of the manual over-ride switch, described in d above (including the ability to resume operation with a second toggle).

Input summary: 1 momentary switch (manual over-ride)
Output summary: 1 LED (warning light)
2 7-segment displays (radiation count)

5b. Automatic Geiger Counter (Enhanced Display) (Difficulty: 7)

For easy viewing, there is also a bar graph of 9 LEDs displaying the activity. Each step on the bar graph represents 10 counts: 0–9 counts lights nothing, 10–19 counts lights one LED, 20-29 counts lights two LEDs, etc. This is active during both the accumulation and the display phase.

Additional Output summary: 10 LEDs (bar graph; chip-sized packages with 8 onboard LEDs are available)

6. Drunk Driver Control (Difficulty: 8)

A car has a circuit to prevent drunk driving. When a user inserts a key into the ignition and turns it, two 1-digit numbers appear on separate seven-segment displays. In addition, a timer begins counting down on a third seven-segment display. The user has 9 seconds to input the absolute value of the difference between the two values displayed.

a. See the end of this document for the generation of random numbers. It should be possible for either of the random number displays to be the largest.
b. The current value of the user input will be displayed on a fourth seven-segment display. A toggle switch will be used to increment this value, and a second toggle switch to enter this value.
c. If the correct value is entered, then an LED will light to indicate ignition.
d. If the timer runs out (no value is entered), or if an incorrect value is entered, then a warning LED must go on. This LED must stay on for ~10 minutes, during which time the ignition switch should have no effect.
e. When the key is removed after ignition, the ignition light should go off and the system reset.
f. However, if the key is removed during the entry window or after failure, the event sequence should not be interrupted.

NOTE: Yes, the warning LED must really light for a full 10 minutes. During development and testing, you may shorten this to 6 seconds by increasing the clock rate by a factor of 100.

Input summary: 1 double-throw switch (indicates key in ignition)
2 momentary switches (increment, and enter)
Output summary: 2 LEDs (incorrect value/wait 10 minutes, and ignition)
4 7-segment displays (2 test values, user input value, and timer)

6b. Enhanced Drunk Driver Control (Difficulty: 10)

The same operation using two digit numbers, shown on three pairs of 7-segment displays.

a. See the end of this document for the generation of random numbers.
b. A toggle switch will be used to increment the ones place of the user entry, and a separate toggle to increment the tens place.
c. If the key is not removed during the 10 minute warning period, then the warning period should continue until the key is removed.

Additional Input summary: 1 momentary switches (increment tens)
Additional Output summary: 3 7-segment displays (tens place for 2 test values and user input value)

7. Stopwatch (Difficulty 9)

A stopwatch is desired that can be used in "count up" mode for things like track meets, but in "count down" mode for things like an egg timer. The output of the stopwatch should be in the form "###.#" (with appropriate blanking). All features of the stopwatch must operate using only two buttons (i.e., momentary switches). One of the buttons will act as a "mode select" button. The other "operate" button will cause appropriate actions for each mode. Features must include:

a. While in "normal" mode, the second button acts as a start/stop/continue button.
b. The user can select whether this normal operation is in the "count up" or "count down" direction.
c. Count down direction always stops and holds at zero. Two project options are permitted: If switching to count up mode at that point results in the stopwatch counting up, then an additional "timer active" LED must be included, showing the start/stop status. If switching to count up mode at that point results in the stopwatch holding at zero, no extra LED is needed.

d. The user can load a 3-digit value into memory, representing seconds (no tenths). The same 7-segment displays must be used to show the value being loaded. When the user enters the "load memory" mode, it starts with the previous memorized value.
e. There must be a way to reset the timer. While the direction is count up, this sets the timer to zero. While the direction is count down, this sets the time to the value loaded in memory.
f. The user can select between "alarm on" and "alarm off" modes, indicated by an LED. Use a second LED to indicate when the alarm is sounding. With count down direction, the alarm sounds when zero is reached. With count up direction, the alarm sounds when the loaded value is reached. The alarm is active until the user turns it off. When the alarm becomes active, the system mode should automatically be changed to "shut off alarm" mode.
g. An LED panel must indicate the currently selected mode.
h. Since there is some flexibility in how you meet these requirements, you must provide the instructor with an instruction "manual" so that all features can be verified. Score will be lower if the instructor cannot figure out operation based on manual alone.

Input summary: 2 momentary switches (mode select and mode action)
Output summary: multiple LED panel (mode indicator; chip-sized packages with 8 LEDs are available)
2 LEDs (alarm on, and alarm sounding; buzzers are available for realistic behavior)
1 LED (timer active) *if* count down to zero does not turn off timer.
4 7-segment displays (current time/alarm time)

7b. Enhanced Stopwatch (Improved Loading) (Difficulty: 10)

For loading a large value into memory, there must be a more convenient option than counting up to the desired value. Suggested possibilities are:

a. Count up "warp": If the "operate" button is held for a certain time, counting speeds up by a factor near 10x.
b. Set each digit of the load value separately.

No Input/Output changes

8. Package Sorter (Difficulty: 8)

A mail sorting system has deflector gates along a linear conveyor belt, having stations "ABCDE". While operating, a new package enters the system at point "A" each second; each package has a two bit bar code, indicating its final destination. The bar code is read at point "A" on the conveyor belt. After one second passes, each package moves to the right one station (e.g., a package at C moves to D).

a. The output will consist of 5 seven-segment displays, one for each station. Once the system is started, each display indicates the value of the bar code of the package currently at that station. If there is no package at any particular station, then the display there should be blank.
b. The system must start up indicating that all stations are empty.
c. If a package has the barcode 0_{10} (00_2), it is deflected from the conveyor belt at B. Whenever this happens, a corresponding LED must light up for ½ second to indicate a deflection occurring there. Note that stations CDE will subsequently be empty, as that position moves down the belt.

d. Similarly, packages with barcode 1_{10} should deflect at station C, barcode 2_{10} should deflect at D, and barcode 3_{10} should deflect at E.
e. Packages arrive with random bar codes. See the end of this document concerning random numbers.
f. A switch controls whether the conveyor belt is running or stopped. While stopped, a second switch may be toggled to clear all packages from all stations.

An example output over a few seconds is provided to the right. The first row would occur immediately after start up, or after toggling the clear switch. A "*" indicates a deflection LED being temporarily lit.

Hint for determining when blanking is appropriate: note that no package of type 0_{10} will ever reach station C.

Input summary: 1 double-throw switch (run/stop)
1 momentary switch (clear)
Output summary: 5 7-segment displays (bar codes)
4 LEDs (deflection indicator)

A	B(0)	C(1)	D(2)	E(3)
2				
0	2			
1	0*	2		
2	1		2*	
3	2	1*		
1	3	2		
1	1	3	2*	
0	1	1*	3	
2	0*	1*		3*
2	2			
1	2	2		
3	1	2	2*	

9. Combination Lock (Difficulty: 7)

A security lock has 4 buttons. Four buttons must be pressed in the proper sequence, with repeats possible, to open the lock (e.g., ACDB or BBAB or DDAA).

a. The entry code must be programmable using switches. Two switches are needed to represent each digit of the sequence, for a total of 8 switches. In a real application, these switches would be hidden from the person trying to open the lock.
b. There must be a clear switch to restart the sequence if the user accidentally presses a wrong button.
c. If the sequence is incorrect after 4 switches have been pressed, then all further attempts must be locked out, and an alarm activated. The alarm must stay active until the lock is reset.
d. Once the lock is opened, it remains open for 3 or 4 seconds. After this, the system is reset (to the same combination) for the next user.
e. The system must initialize to a pre-determined state when initially powered.

Input summary: 8 double-throw switches (code programming; chip-sized packages of switches, as well as groups of resistors, are available)
6 momentary switches (4 for code entry, clear, and reset alarm)
Output summary: 2 LEDs (open, and alarm)

9b. Combination Lock (with Program Mode) (Difficulty: 8)

Instead of programming the entry code by means of 8 switches, there is a single switch (in a secure area) that controls whether the system is in "program" mode or "entry" mode.

a. While in "program" mode, pushing the four buttons in sequence sets the entry code. The last four buttons pushed, in the order entered, are the entry code. Implementing a reset during programming is optional. You should clearly define the behavior if fewer than 4 buttons are pushed before leaving program mode.
b. While in "entry" mode, behavior is the same as in the unmodified project

***Removed* Input summary**: 8 double-throw switches (code programming)
Additional Input summary: 1 double-throw switch (mode)

10. Serial Data Buffer (Difficulty: 7)

A 6 bit stream of bits is generated in one register, then moved one bit at a time (at ~1 Hz) over a serial wire, to a receiving register. Five of the 6 bits (A_0 through A_4) represent data, and the final bit (A_5) is a "validate" bit. The validate bit is automatically generated so that there is an even number of 1's in the 6 bits sent. Due to noise, any *one* of the 6 bits being transferred across the serial wire may become corrupted (inverted). We'll assume that the noise is never so bad that two bits get corrupted.

a. A "register panel" displays the contents of each of a source register and a receive register. The panel consists of 6 LEDs (including the validate bit) and a pair of 7-segment display showing the value of the data bits only.
b. The user must set the original 5 data bits using 5 switches. While the data is being adjusted, the source register panel must be continuously updated.
c. The receive register and its panel must be on an external board, and the function of the wires going to that board must be readily apparent. The receive panel must be initialized blank on power up.
d. When a send button is pushed, the receive register panel must go blank (if a previous transmission is showing) and then data transfer must start. The user must be unable to change the original values until transfer is complete, and the source register panel should not change. A "sending" LED must be lit during transfer.
e. Bits are transferred one at a time at ~1 Hz, starting with the LSB. The receive register is a shift register, loading through the MSB (see sample output). Receive register LEDs should light or not, as appropriate, as each bit is transferred.
f. When the transfer is complete, the 7-segment displays of the receive register panel should indicate the value of the received data. This, along with the LEDs, should remain unchanged until the next time the send button is pressed.
g. There must be a switch for the instructor to introduce "noise" (that is, invert the transmitted bit) during the transfer process. No more than one bit per transfer will be corrupted during testing.
h. If the signal is corrupt, a warning light should come on. Hint: remember that a multiple input XOR gate can be used as a parity checker.
i. If the received message is corrupt, after about 1 second the message should be resent, as if the send button had been pressed again.
j. Once all bits are successfully sent, the user can then reset the 5 data switches, and send another set.

The following table illustrates what the LEDs might look like after the send button is pushed. No noise error is introduced in this example. A is the source panel, and B is the receive panel.

A_5	A_4	A_3	A_2	A_1	A_0		B_5	B_4	B_3	B_2	B_1	B_0
1	0	1	0	1	1							
1	0	1	0	1	1		1					
1	0	1	0	1	1		1	1				
1	0	1	0	1	1		0	1	1			
1	0	1	0	1	1		1	0	1	1		
1	0	1	0	1	1		0	1	0	1	1	
1	0	1	0	1	1		1	0	1	0	1	1

Input summary: 6 double-throw switches (5 for data, 1 for noise activation)
1 momentary switch (send)

Output summary: 12 LEDs (6 for original data, 6 for received data; chip-sized packages with 8 onboard LEDs are available)
2 LEDs (sending and corrupt)
4 7-segment displays (two pairs, source data and received data)

11. Adding Machine (Difficulty: 11)

(Simpler versions available)

An adding machine adds or subtracts a one-digit number to a bank that can range between −99 and 99. All three digits are shown on 7-segment displays. To operate, a momentary switch is toggled until the desired one digit input is displayed. Then, an "add" or a "subtract" button is pushed, and the result is shown on a two digit display, replacing the previous display. You should consider using "twos complement" numbers for this project. This format, as well as addition and subtraction, will be covered in lecture shortly after the final projects are started.

a. All displays must initialize to zero. Also, there must be a "clear" switch to reset the system to the initial state.
b. There must be a way to quickly preset the 2-digit display to any desired value; a set of switches representing the desired value in any convenient format may be used along with a load button.
c. There must be an "overflow" LED when the result is greater than 99 or less than –99. Once overflow occurs, no more additions, subtractions, or loads are allowed.
d. There must be a "negative" LED to represent negative numbers (a minus sign), for use when the result is less than zero. This should be suitably placed next to the bank.

Note that, since zero can be loaded into the bank, the purpose of the "clear" switch is to recover from an overflow.

Input summary: 8 double-throw switches (number to load)
5 momentary switches (increment input, add, subtract, clear, and load)
Output summary: 2 LEDs (overflow and negative bank)
3 7-segment displays (1 for input, 2 for bank)

11b. Adding Machine (Positive Bank) (Difficulty: 7)

The bank need only handle positive numbers: 0 through 99. You should **not** use "twos complement" numbers for this project.

***Removed* Output summary**: 1 LED (negative bank)

11c. Adding Machine (Octal Bank) (Difficulty: 9)

All numbers in the project are displayed in octal format. Bank can range from -77_8 to 77_8, and one can add or subtract from 0 to 7.

***Modified* Input summary**: 6 double-throw switches (number to load)

12. Chip Tester (Difficulty: 7)

The circuit must determine whether all 4 gates on three specific types of quad-gate chips work (choose types from 7400, 7408, 7432, or 7486). There is one open "bay" where the chip to be tested will be placed. To operate, the user places a chip into this bay. The user specifies which type of chip it is by throwing one of four SPDT switches (you may assume that only one switch is thrown at a time). Then, the user presses a momentary "start" switch. If all 4 gates on the chip work perfectly, for all 4 possible input conditions, then an "OK" light comes on. If any of the 4 gates fails to function for any of the 4 possible input combinations, then a "BAD chip" light comes on. An

additional 4 warning lights indicate which, if any, of the gates are bad. All LEDs remain lit until the next test is initiated by the user again pressing the "start" switch. The system powers up with all LEDs off, and can be reset to that state by a second momentary switch.

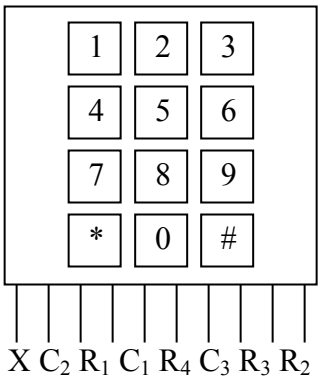
Input summary: 4 switches (choose test type)
2 momentary switches (start, clear)
1 bay (for chip to be tested)

Output summary: 6 LEDs (OK chip, bad chip, and 4 gate-specific warnings)

13. Telephone Dialer (Difficulty: 11)

Telephone keypads have 12 keys, arranged in a grid. The keypad has 7 wires connected to it. When a key is pressed, two of the wires are connected together. For example, when the "8" is pressed, wires R_3 ("row 3") and C_2 ("column 2") are connected together. When the user presses 7 numerical keys in any combination, the values of these keys should be displayed in order by a group of 7-segment displays.

a. All displays must initialize blank.
b. The first digit pressed should be displayed in the MSD, the seventh in the LSD.
c. There should be a reset switch (hang up).

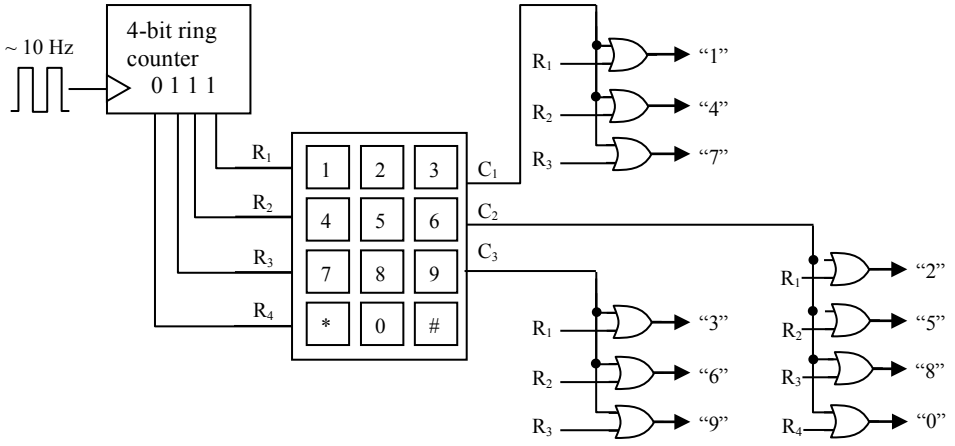


Unit layout

Input summary: 1 12-button keypad
1 momentary switch (reset)

Output summary: 7 7-segment displays

An active-low suggestion for use of the keypad is provided. Note that the circuit fragment shown has a problem with repeating digits that needs to be corrected.



13b. Telephone Dialer (One Digit) (Difficulty: 9)

Use only one 7-segment display, which should initialize blank. Thereafter, it should display the last number pressed. Use the funny shapes which our displays show for 1010 and 1110 to indicate the * and # keys respectively.

In addition, have an LED which lights while any key is being pressed, and one 7-segment display which shows how many times a button has been pressed since the project was turned on. When this display reaches 9, it should roll over to 0 with the next button press.

***Removed* Input summary**: 1 momentary switch (reset)

***Modified* Output summary**: 2 7-segment displays (last key, and number of presses)
1 LED (key pressed)

14. Binary to Decimal Converter (Difficulty: 9)

The binary to decimal converter takes an 8 bit binary number and converts it to decimal form. The user inputs an unsigned binary number by setting eight double throw switches. Calculation takes place each time a momentary switch is thrown. The output is shown in decimal using three 7-segment displays.

a. All displays must initialize blank.
b. The first two digits should be blank as appropriate.
c. Once the user changes any part of the binary number by throwing a switch, the display should blank until the load switch is pressed again.

Input summary: 8 double throw switches (number to load)
1 momentary switch (load)
Output summary: 3 7-segment display (result of previous load)

15. Combat Simulation Game (Difficulty: 10)

There is a single score that indicates the current state of the game. The score begins at 7, and the game ends either when the score reaches 14 (player wins), or when it reaches 0 (player loses). The user selects an action (sword attack, sword defense, magical attack, or magical defense), and then presses "go". The circuit similarly chooses (randomly) one of these 4 choices. The score is adjusted based on the following chart.

		Player Choice			
		attack	defend	magic attack	magic defense
Machine Decision		0	1	2	3
Attack	0	0	0	+1	-1
Defend	1	0	0	+2	-2
Magic Attack	2	-1	-2	0	+3
Magic Defense	3	+1	+2	-3	0

a. The current score is displayed on a bar chart of 14 LED's.
b. A 7-segment display will show the most recent choice of the machine player
c. A yellow LED will light if the score reaches 0 (player loses).
d. A green LED will light if the score reaches 14 (player wins).
e. Pressing reset returns the score to 7, and turns off both the "win" and "lose" LED's. It also blanks the 7-segment display.

Input summary: 2 double throw switches (to choose actions 0, 1, 2, or 3)
2 momentary switches (go, restart)
Output summary: 14 LED's (score; chip-sized packages with 8 LEDs are available)
2 LED's (win and loss)
1 7-segment display (machine action indicator)

Projects Designed & Rated including a BASIC Stamp

16. Battleship (Difficulty: 9)

The BASIC Stamp generates an ocean (a 5 × 5) grid of spaces, into which it randomly places two ships, one of which is 2 spaces long, and the other is 3 spaces long. The ships are randomly oriented horizontally or vertically. The player has 12 shots with which to sink both ships. Remaining ammunition is shown on a pair of 7 segment displays. Double throw switches or potentiometers are

used to select the row and column for the player's next shot (values 1 through 5 shown on another pair of 7 segment displays). A momentary switch is used to confirm/fire the shot. After each shot, the ammo remaining is reduced by one. If the shot is a hit, an LED activates for 1 second. If the shot is a miss, a different LED lights for one second. The user must keep track of which destinations have already been fired upon. Two LEDs light when a ship has been sunk (one LED per ship) and remain lit for the remainder of the game. There are a "game won" LED (if both ships are sunk before the ammunition is depleted) and a "game lost" LED (if one or more ships survive and the ammo runs out). Finally, there should be a game reset/start switch. This switch will also be used to initialize the random number generator (see end of this document).

Input summary: 2 momentary switches (fire, start/restart game)
2 potentiometers or 10 double throw switches (row/column selection)

Output summary: 6 LEDs (hit, miss, win, lose, sunk A, sunk B)
4 7-segment displays (row, column, 2 for ammo)

17. Slot Machine (Difficulty: 8)

Using a potentiometer, the user will place a bet ranging from 1-3 credits. The result is displayed on a 7 segment display. Once a desired bet is selected, the machine is activated by a momentary switch. Three separate 7 segment displays will scroll through numbers from 0-3 at a moderate frequency (around 5 to 10 Hz). These 7 segment displays will stop scrolling in sequential order; the first stopping after 3 seconds, the second after 4 seconds and the third after 5 seconds. During this time, the user may not change the bet or restart. A winning spin occurs when all three displays end with the same number. Payouts are multiples of the bet, as follows: 000 → 5 times, 111 → 10 times, 222 → 15 times, 333 → 30 times. (jackpot!) After the spin completes, an appropriate sound is played, and a display shows the payout (or 0 for a losing spin).

Input Summary: 1 momentary switch (pull lever/start)
1 potentiometer (wager)

Output Summary: 6 7-segment displays(1 for wager, 3 for the symbols, 2 for payoff)

18. Switch-a-mole (Difficulty: 8)

This is a game similar to whack-a-mole. After a momentary start switch is pressed, one of 6 LEDs will light for a 3 second time period. Each of the 6 LED's has a corresponding button. The user must press the correct button before the light turns off, or the game ends. If the user presses the correct button, immediately that light turns off. As the game proceeds, the duration allowed for each mole gradually gets shorter. A pair of 7-segment displays will show the number of buttons that have been correctly pressed since the start button was pressed.

The second mole will appear 3 seconds after the first. As the game proceeds, the time between new moles appearing also gets gradually shorter; that change is more rapid than the time allowed for each mole, so that it becomes possible to have more than one "mole" can be active at a time. The way in which these two times shrink must never result in times shorter than 0.3 seconds, and must be more complicated than a linear change to that threshold. The game parameters must be adjusted so that typical play results in scores of around 25, with at least 2 moles simultaneously active by the end.

Input Summary: 1 momentary switch (start, see end of this document concerning random numbers)
6 momentary buttons (whack)

Output Summary: 6 LEDs (moles)
2 7-segment displays (score)

19. Simon Memory Game (Difficulty: 7)

A random array having values from 0 to 3 is generated, with a maximum length 32. Each value is represented by a unique colored LED, and by a unique musical tone. Each value also has a corresponding button. The tone and light for the array elements are displayed sequentially, each for 0.5 seconds with a noticeable pause in between (to make repeated array elements easier to detect). The player then presses the corresponding buttons to repeat the pattern. This also lights the LEDs and plays the tone for 0.5 seconds each (although the buttons may be pressed more quickly than that).

The length of the array is initially 1. Each time the array is correctly repeated by the player, an element is added to the array and the sequence is repeated. When the player makes a mistake, the game is over, and the number of correct responses is displayed for the score.

Input Summary: 1 momentary switch (start, see end of this document concerning random numbers)
4 momentary buttons

Output Summary: 4 LEDs with different colors (array elements)
1 speaker
2 7-segment displays (score)

20. Push Push Revolution (Difficulty: 6)

A random value ranging from 0 to 3 is generated. Each value is represented by a unique colored LED, and by a unique musical tone. The light is displayed for 2 seconds, and the tone is sounded for two seconds starting 1 second after the light turns on. The player must press the corresponding button while the tone is sounding. If incorrect or too late, the game is over.

This cycle is repeated, with the time allowed getting gradually shorter to a minimum of 0.4 seconds. Each new light is displayed immediately following the previous one, while the previous tone is still being sounded. The change in tempo must be gradual enough so that the rhythm of the cycles remains easy to detect, but rapid enough so that a typical game does not take more than a few minutes. As the game progresses, two 7-segment displays show the number of correct presses so far.

Input Summary: 1 momentary switch (start, see end of this document concerning random numbers)
4 momentary buttons

Output Summary: 4 LEDs with different colors (array elements)
1 speaker
2 7-segment displays (score)

21. Mini-synthesizer (Difficulty: 9)

Eight buttons act as the keys of a keyboard. When each is pressed, a tone is sounded, with frequencies of a major scale. One turn of a potentiometer adjusts the frequency of the lowest note between 220 Hz and 440 Hz. In "free play" mode, each tone lasts as long as the key is depressed, subject to a minimum of 0.1 s. If more than one key is pressed at a time, the result is not specified here, but you must be able to describe the result for your project.

A switch turns off and on "tempo control" mode. In this mode, all notes last for a specified duration between 0.3 s and 2 s, as controlled by a second potentiometer. If a new key is pressed while a tone is sounding, the corresponding tone is played in the next time interval. If multiple keys are pressed while a tone is sounding, the result is not specified here, but you must be able to describe the result for your project.

Input Summary: 8 momentary buttons (tone keys)
2 potentiometers (tuner, tempo control)
1 double throw switch (mode select)
Output Summary: 1 speaker

Generating "Random" Numbers

… with TTL

Connecting a 555 chip running at a high frequency to a counting register can generate a random number. At the moment when a number is desired, either block the counting clock signal, or copy the counting register into another register. As long as the timing is much slower and unrelated to the frequency of the 555 chip, the result will seem random. Choose an appropriate counter for the specific project (e.g., 74192 decimal counters for a decimal random number).

… with the BASIC Stamp

PBASIC has a 'RANDOM' statement that can be used to generate pseudo-random 16-bit values. RANDOM uses a sequence of the 65536 possible 16-bit numbers that has no apparent order. By giving RANDOM a "seed," you choose a starting point in this sequence, and each call of the RANDOM statement returns the next number in the sequence. However, starting with the same seed will always produce the same sequence of numbers.
In order to start at a random point in the pseudo-random sequence, call the RANDOM statement repeatedly until the user presses a button. This will run through a large number of (unused) steps in the sequence, so that the starting point is different each time the program is run.

```
randval VAR word
randval = 11000          'initialize to arbitrary for added randomness
DO
     RANDOM randval
LOOP UNTIL IN1 = 0       'user button connected to PIN 1
```

Any piece of the resulting number can be used as a random number with a smaller range. For instance, you could use a subset of its bits, or use integer division.

```
newvar VAR nibble
newvar2 VAR byte
newvar = randval & 0xF     'lowest 4 bits, range 0 to 15
newvar2 = randval//6             'modulus operator, range 0 to 5
```

Each call to RANDOM will put a new random number in the variable. After initialization, the variable should not be assigned to, since that would put us back at a specific point in the pseudo-random sequence (likely the same each time the program runs).