Projects for Join us on an ‘electrifying’ adventure with Snap Circuits! Presentation

Learning Objective 1 Projects

Show how to do basic electronics with the Jr. kit and Bric for more creativity.

Project 41: Lamp & Fan in Series

WARNING: Moving parts. Do not touch the fan or motor during operation. Do not lean over the motor.

Turn on the slide switch (S1). The lamp (L1) lights up, and the motor (M1) spins the fan. Notice how the lamp gets a little less bright as the motor speeds up.

The faster the motor is spinning, the less electricity it needs. The more electricity flows, the brighter the lamp gets. The motor needs the most electricity when it starts up, making the lamp brightest. Without the fan, the motor can spin fast and needs little electricity, making the lamp dim.

Parts needed

Qty	Part
1	(B1) Battery holder
1	(S1) Slide Switch
1	(L1) Lamp
1	(M1) Motor
1	Fan

First Layer

1. (B1) Battery holder on B3 and GND on D3
2. (L1) Lamp on B1 - D1

Second Layer

3. (M1) Motor on B1 - B3 with positive on B3
4. (S1) Slide Switch on D1 - D3

Third Layer

5. Fan on (M1) Motor

Project 43: Lamp & Fan in Parallel

WARNING: Moving parts. Do not touch the fan or motor during operation. Do not lean over the motor.

Turn on the slide switch (S1). The lamp (L1) lights up, and the motor (M1) spins the fan.

Compare this circuit to the circuit in project 41, and also try removing the fan as done in project 42. Notice how the lamp brightness is not affected by the motor speed, and the motor starts a little faster.

Here the motor and lamp are connected in parallel. Each has its own path to the batteries, so they don't affect each other.

An advantage of connecting parts in parallel is that if one of them burns out, the other will still work. The switch is connected in series with both the lamp and motor, so if it breaks, nothing will work. Electricity flows out of the batteries, through either the motor or lamp, then back to the batteries through the switch.

Accessibility Note - For those who are blind or low vision, use (W1) Horn instead of the (L1) Lamp piece.

Parts needed

Qty	Part
1	(B1) Battery holder
3	3-Snap Wire
1	(S1) Slide Switch
1	(L1) Lamp
1	(M1) Motor
1	Fan

First Layer

1. (B1) Battery holder on A6 and GND on C6
2. 3-Snap Wire on A2 - A4
3. 3-Snap Wire on C2 - C4

Second Layer

4. M1) Motor on A2 - C2 with positive on C2
5. L1) Lamp on A3 - C3
6. S1) Slide Switch on C4 - C6
7. 3-Snap Wire on A4 - A6

Third Layer

8. Fan on (M1) Motor

Project 11: *LED IN SERIES & PARALLEL*

Build the circuit as described below and turn on the slide switch (S1). The LEDs (D8, D11, & D12) light; watch how their patterns change.
Try swapping the LED locations and replacing one with the blue LED (D9). Try all combinations and see how the effects change. You can also place the unused 3-snap wire where the color2 LED (D12) is and see how the circuit changes.

Parts Needed

Qty	Part
6	Bric2snap Adapter 2x2
1	Plate 2x2
1	(S1) Slide Switch
2	3-snap wire
1	(D8) Color LED
2	(D11) Blink Red LED
1	(D12) Color2 LED
1	(B3) Battery Holder

Bric Layer 1

1. Place a Bric2snap Adapter 2x2 on C3-D4
2. Place a Bric2snap Adapter 2x2 on C10-D11
3. Place a Bric2snap Adapter 2x2 on C17-D18
4. Place a Bric2snap Adapter 2x2 on J3-K4
5. Place a Bric2snap Adapter 2x2 on J10-K11
6. Place a Bric2snap Adapter 2x2 on J17-K18
7. Place a Plate 2x2 on F24-G25

First Layer

8. Place the Battery Holder (B3) on the adapter C17-D18 and on the adapter J17-K18 with the positive on the C17-D18 adapter
9. Place a 3-Snap Wire on the adapter C3-D4 and C10-D11
10. Place a 3-Snap Wire on the adapter J3-K4 and J10-K11

Second Layer

11. Place the Slide Switch (S1) on the 3-Snap Wire on K10-L11 and on the negative side of the Battery Holder (B3) on K18-L19
12. Place the Color2 LED (D12) on the 3-Snap Wire on C10-D11 and on the positive side of the Battery Holder (B3) at C18-D19 with the positive side of the LED on the Battery Holder (B3)
13. Place the Blink Red LED (D11) on the middle of both 3-Snap Wires at C6-D7 with the positive side on C6-D7
14. Place the Color LED (D8) on the end of each 3-Snap Wires at C2-D3 and K2-L3 with the positive side on C2-D3

Here the color LED and blink red LED are connected in parallel and then connected in series with the color2 LED to produce some interesting effects. The electricity from the batteries flows through the color2 LED, then splits up between the color and blink red LEDs, then re-combines in the switch before returning to the batteries.
When LEDs are connected in series, the battery voltage may not be strong enough to fully turn them on. Red light is easier to produce then the other colors and so turns on more easily.

Learning Objective 2 Projects

Take a project from the Snapino kit and make it into a standalone talking light meter using parts from both the Snap Circuit Jr and Snapino kits.

Project 11 - Light Monitor

Parts Needed

Qty	Part
1	Snapino module with connection cable
1	(R4) Resistor 10kΩ
1	(Q4) Phototransistor
1	White jumper cables
1	2-Snap wire
1	3-Snap wire
1	5-Snap wire

First Layer

1. Snapino module 5V on A2, GND on C2, and D11 on C5
2. Phototransistor Q4 on A1 - C1
3. 5-Snap wire on E1 - E5

Second Layer

4. Resistor on C1 - E1
5. One side of white jumper cable B2
6. 2-Snap wire on A1 - A2
7. 3-Snap wire on C2 - E2

Third Layer

8. One side of white jumper cable C1

Light monitor code

```
//Light_Monitor

const int phototransistor = A0;
int Val = 0;
```

```
void setup() {
 Serial.begin(9600);                        //Open terminal window
 pinMode(phototransistor, INPUT);
}

void loop() {
 Val = analogRead(A0);                      //Measure light value

 Serial.print("Measured light value = ");  //Print text
 Serial.println(Val);                       //Print light value
 delay(250);
}
```

This project opens a window on your computer to display measured data in real time.

Build the circuit shown. Load the sketch Light Monitor code above into Snapino using the programming instructions in Project 3. When the upload is finished, click on the Tools menu, then pick Serial Monitor from the list. A window opens on your computer screen, displaying the measured light value. Vary the amount of light shining on the phototransistor (Q4) and see how the number displayed changes.

Programming Note: The **Serial** commands open the window then display some text with the light measurement.

Changing the amount of light shining on the phototransistor changes its resistance, and so it changes the voltage measured at the A0 input. The **analogRead** command has 10-bit accuracy, so the measured number is from 0 (darkest) to 1023 (brightest).

The 10kΩ resistor (R4) allows the voltage at the A0 input to fall when it is dark and rise when there is light on the phototransistor. The voltage measured depends on the ratio of the phototransistor resistance to the 10kΩ resistor (R4). The measured value is about 512 when the phototransistor resistance equals R4. Replacing R4 with another resistor would shift the measured light value.

Parts Needed for self voicing light monitor

Qty	Part
1	Snap to pin connector from Snapino
1	(SP1) Speaker from Jr kit

Additional instructions.

1. Snap to pin jumper from D3 header to e7
2. (SP1) Speaker on E5-E7

Self Voicing light monitor

```
//Light_Monitor

/*
* Talkie library
* Copyright 2011 Peter Knight
```

```
*
* This code is released under GPLv2 license.
*
* Now for something a bit more challenging.
*
* Building sentences by program.
*
* The sayNumber() function can say any number under a million by
* building the number from short phrases,
*
* Connect       a sensor to Analog 0, and this program will read the sensor
voltage.
*/

#include "talkie.h"

Talkie voice;

const uint8_t spZERO[] PROGMEM = {
    0x69, 0xFB, 0x59, 0xDD, 0x51, 0xD5, 0xD7, 0xB5, 0x6F, 0x0A, 0x78, 0xC0,
    0x52, 0x01, 0x0F, 0x50, 0xAC, 0xF6, 0xA8, 0x16, 0x15, 0xF2, 0x7B, 0xEA,
0x19, 0x47, 0xD0, 0x64, 0xEB, 0xAD,
    0x76, 0xB5, 0xEB, 0xD1, 0x96, 0x24, 0x6E, 0x62, 0x6D, 0x5B, 0x1F, 0x0A,
0xA7, 0xB9, 0xC5, 0xAB, 0xFD, 0x1A,
    0x62, 0xF0, 0xF0, 0xE2, 0x6C, 0x73, 0x1C, 0x73, 0x52, 0x1D, 0x19, 0x94,
0x6F, 0xCE, 0x7D, 0xED, 0x6B, 0xD9,
    0x82, 0xDC, 0x48, 0xC7, 0x2E, 0x71, 0x8B, 0xBB, 0xDF, 0xFF, 0x1F
};

const uint8_t spONE[] PROGMEM = {
    0x66, 0x4E, 0xA8, 0x7A, 0x8D, 0xED, 0xC4, 0xB5, 0xCD, 0x89, 0xD4, 0xBC,
    0xA2, 0xDB, 0xD1, 0x27, 0xBE, 0x33, 0x4C, 0xD9, 0x4F, 0x9B, 0x4D, 0x57,
0x8A, 0x76, 0xBE, 0xF5, 0xA9,
    0xAA, 0x2E, 0x4F, 0xD5, 0xCD, 0xB7, 0xD9, 0x43, 0x5B, 0x87, 0x13, 0x4C,
0x0D, 0xA7, 0x75, 0xAB, 0x7B,
    0x3E, 0xE3, 0x19, 0x6F, 0x7F, 0xA7, 0xA7, 0xF9, 0xD0, 0x30, 0x5B, 0x1D,
0x9E, 0x9A, 0x34, 0x44, 0xBC,
    0xB6, 0x7D, 0xFE, 0x1F
};

const uint8_t spTWO[] PROGMEM = {
    0x06, 0xB8, 0x59, 0x34, 0x00, 0x27, 0xD6, 0x38, 0x60, 0x58, 0xD3, 0x91,
    0x55, 0x2D, 0xAA, 0x65, 0x9D, 0x4F, 0xD1, 0xB8, 0x39, 0x17, 0x67, 0xBF,
0xC5, 0xAE, 0x5A, 0x1D, 0xB5,
    0x7A, 0x06, 0xF6, 0xA9, 0x7D, 0x9D, 0xD2, 0x6C, 0x55, 0xA5, 0x26, 0x75,
0xC9, 0x9B, 0xDF, 0xFC, 0x6E,
    0x0E, 0x63, 0x3A, 0x34, 0x70, 0xAF, 0x3E, 0xFF, 0x1F
};

const uint8_t spTHREE[] PROGMEM = {
    0x0C, 0xE8, 0x2E, 0x94, 0x01, 0x4D, 0xBA, 0x4A, 0x40, 0x03, 0x16, 0x68,
    0x69, 0x36, 0x1C, 0xE9, 0xBA, 0xB8, 0xE5, 0x39, 0x70, 0x72, 0x84, 0xDB,
0x51, 0xA4, 0xA8, 0x4E, 0xA3, 0xC9,
    0x77, 0xB1, 0xCA, 0xD6, 0x52, 0xA8, 0x71, 0xED, 0x2A, 0x7B, 0x4B, 0xA6,
0xE0, 0x37, 0xB7, 0x5A, 0xDD, 0x48,
    0x8E, 0x94, 0xF1, 0x64, 0xCE, 0x6D, 0x19, 0x55, 0x91, 0xBC, 0x6E, 0xD7,
0xAD, 0x1E, 0xF5, 0xAA, 0x77, 0x7A,
```

```
    0xC6, 0x70, 0x22, 0xCD, 0xC7, 0xF9, 0x89, 0xCF, 0xFF, 0x03
};

const uint8_t spFOUR[] PROGMEM = {
    0x08, 0x68, 0x21, 0x0D, 0x03, 0x04, 0x28, 0xCE, 0x92, 0x03, 0x23, 0x4A,
    0xCA, 0xA6, 0x1C, 0xDA, 0xAD, 0xB4, 0x70, 0xED, 0x19, 0x64, 0xB7, 0xD3,
0x91, 0x45, 0x51, 0x35, 0x89,
    0xEA, 0x66, 0xDE, 0xEA, 0xE0, 0xAB, 0xD3, 0x29, 0x4F, 0x1F, 0xFA, 0x52,
0xF6, 0x90, 0x52, 0x3B, 0x25,
    0x7F, 0xDD, 0xCB, 0x9D, 0x72, 0x72, 0x8C, 0x79, 0xCB, 0x6F, 0xFA, 0xD2,
0x10, 0x9E, 0xB4, 0x2C, 0xE1,
    0x4F, 0x25, 0x70, 0x3A, 0xDC, 0xBA, 0x2F, 0x6F, 0xC1, 0x75, 0xCB, 0xF2,
0xFF
};

const uint8_t spFIVE[] PROGMEM = {
    0x08, 0x68, 0x4E, 0x9D, 0x02, 0x1C, 0x60, 0xC0, 0x8C, 0x69, 0x12, 0xB0,
    0xC0, 0x28, 0xAB, 0x8C, 0x9C, 0xC0, 0x2D, 0xBB, 0x38, 0x79, 0x31, 0x15,
0xA3, 0xB6, 0xE4, 0x16, 0xB7,
    0xDC, 0xF5, 0x6E, 0x57, 0xDF, 0x54, 0x5B, 0x85, 0xBE, 0xD9, 0xE3, 0x5C,
0xC6, 0xD6, 0x6D, 0xB1, 0xA5,
    0xBF, 0x99, 0x5B, 0x3B, 0x5A, 0x30, 0x09, 0xAF, 0x2F, 0xED, 0xEC, 0x31,
0xC4, 0x5C, 0xBE, 0xD6, 0x33,
    0xDD, 0xAD, 0x88, 0x87, 0xE2, 0xD2, 0xF2, 0xF4, 0xE0, 0x16, 0x2A, 0xB2,
0xE3, 0x63, 0x1F, 0xF9, 0xF0,
    0xE7, 0xFF, 0x01
};

const uint8_t spSIX[] PROGMEM = {
    0x04, 0xF8, 0xAD, 0x4C, 0x02, 0x16, 0xB0, 0x80, 0x06, 0x56, 0x35, 0x5D,
    0xA8, 0x2A, 0x6D, 0xB9, 0xCD, 0x69, 0xBB, 0x2B, 0x55, 0xB5, 0x2D, 0xB7,
0xDB, 0xFD, 0x9C, 0x0D, 0xD8,
    0x32, 0x8A, 0x7B, 0xBC, 0x02, 0x00, 0x03, 0x0C, 0xB1, 0x2E, 0x80, 0xDF,
0xD2, 0x35, 0x20, 0x01, 0x0E,
    0x60, 0xE0, 0xFF, 0x01
};

const uint8_t spSEVEN[] PROGMEM = {
    0x0C, 0xF8, 0x5E, 0x4C, 0x01, 0xBF, 0x95, 0x7B, 0xC0, 0x02, 0x16, 0xB0,
    0xC0, 0xC8, 0xBA, 0x36, 0x4D, 0xB7, 0x27, 0x37, 0xBB, 0xC5, 0x29, 0xBA,
0x71, 0x6D, 0xB7, 0xB5, 0xAB, 0xA8,
    0xCE, 0xBD, 0xD4, 0xDE, 0xA6, 0xB2, 0x5A, 0xB1, 0x34, 0x6A, 0x1D, 0xA7,
0x35, 0x37, 0xE5, 0x5A, 0xAE, 0x6B,
    0xEE, 0xD2, 0xB6, 0x26, 0x4C, 0x37, 0xF5, 0x4D, 0xB9, 0x9A, 0x34, 0x39,
0xB7, 0xC6, 0xE1, 0x1E, 0x81, 0xD8,
    0xA2, 0xEC, 0xE6, 0xC7, 0x7F, 0xFE, 0xFB, 0x7F
};

const uint8_t spEIGHT[] PROGMEM = {
    0x65, 0x69, 0x89, 0xC5, 0x73, 0x66, 0xDF, 0xE9, 0x8C, 0x33, 0x0E, 0x41,
    0xC6, 0xEA, 0x5B, 0xEF, 0x7A, 0xF5, 0x33, 0x25, 0x50, 0xE5, 0xEA, 0x39,
0xD7, 0xC5, 0x6E, 0x08, 0x14, 0xC1,
    0xDD, 0x45, 0x64, 0x03, 0x00, 0x80, 0x00, 0xAE, 0x70, 0x33, 0xC0, 0x73,
0x33, 0x1A, 0x10, 0x40, 0x8F, 0x2B,
    0x14, 0xF8, 0x7F
};
```

```
const uint8_t spNINE[] PROGMEM = {
    0xE6, 0xA8, 0x1A, 0x35, 0x5D, 0xD6, 0x9A, 0x35, 0x4B, 0x8C, 0x4E, 0x6B,
    0x1A, 0xD6, 0xA6, 0x51, 0xB2, 0xB5, 0xEE, 0x58, 0x9A, 0x13, 0x4F, 0xB5,
0x35, 0x67, 0x68, 0x26, 0x3D,
    0x4D, 0x97, 0x9C, 0xBE, 0xC9, 0x75, 0x2F, 0x6D, 0x7B, 0xBB, 0x5B, 0xDF,
0xFA, 0x36, 0xA7, 0xEF, 0xBA,
    0x25, 0xDA, 0x16, 0xDF, 0x69, 0xAC, 0x23, 0x05, 0x45, 0xF9, 0xAC, 0xB9,
0x8F, 0xA3, 0x97, 0x20, 0x73,
    0x9F, 0x54, 0xCE, 0x1E, 0x45, 0xC2, 0xA2, 0x4E, 0x3E, 0xD3, 0xD5, 0x3D,
0xB1, 0x79, 0x24, 0x0D, 0xD7,
    0x48, 0x4C, 0x6E, 0xE1, 0x2C, 0xDE, 0xFF, 0x0F
};

const uint8_t spTEN[] PROGMEM = {
    0x0E, 0x38, 0x3C, 0x2D, 0x00, 0x5F, 0xB6, 0x19, 0x60, 0xA8, 0x90, 0x93,
    0x36, 0x2B, 0xE2, 0x99, 0xB3, 0x4E, 0xD9, 0x7D, 0x89, 0x85, 0x2F, 0xBE,
0xD5, 0xAD, 0x4F, 0x3F, 0x64,
    0xAB, 0xA4, 0x3E, 0xBA, 0xD3, 0x59, 0x9A, 0x2E, 0x75, 0xD5, 0x39, 0x6D,
0x6B, 0x0A, 0x2D, 0x3C, 0xEC,
    0xE5, 0xDD, 0x1F, 0xFE, 0xB0, 0xE7, 0xFF, 0x03
};

const uint8_t spELEVEN[] PROGMEM = {
    0xA5, 0xEF, 0xD6, 0x50, 0x3B, 0x67, 0x8F, 0xB9, 0x3B, 0x23, 0x49, 0x7F,
    0x33, 0x87, 0x31, 0x0C, 0xE9, 0x22, 0x49, 0x7D, 0x56, 0xDF, 0x69, 0xAA,
0x39, 0x6D, 0x59, 0xDD, 0x82, 0x56,
    0x92, 0xDA, 0xE5, 0x74, 0x9D, 0xA7, 0xA6, 0xD3, 0x9A, 0x53, 0x37, 0x99,
0x56, 0xA6, 0x6F, 0x4F, 0x59, 0x9D,
    0x7B, 0x89, 0x2F, 0xDD, 0xC5, 0x28, 0xAA, 0x15, 0x4B, 0xA3, 0xD6, 0xAE,
0x8C, 0x8A, 0xAD, 0x54, 0x3B, 0xA7,
    0xA9, 0x3B, 0xB3, 0x54, 0x5D, 0x33, 0xE6, 0xA6, 0x5C, 0xCB, 0x75, 0xCD,
0x5E, 0xC6, 0xDA, 0xA4, 0xCA, 0xB9,
    0x35, 0xAE, 0x67, 0xB8, 0x46, 0x40, 0xB6, 0x28, 0xBB, 0xF1, 0xF6, 0xB7,
0xB9, 0x47, 0x20, 0xB6, 0x28, 0xBB,
    0xFF, 0x0F
};

const uint8_t spTWELVE[] PROGMEM = {
    0x09, 0x98, 0xDA, 0x22, 0x01, 0x37, 0x78, 0x1A, 0x20, 0x85, 0xD1, 0x50,
    0x3A, 0x33, 0x11, 0x81, 0x5D, 0x5B, 0x95, 0xD4, 0x44, 0x04, 0x76, 0x9D,
0xD5, 0xA9, 0x3A, 0xAB, 0xF0, 0xA1,
    0x3E, 0xB7, 0xBA, 0xD5, 0xA9, 0x2B, 0xEB, 0xCC, 0xA0, 0x3E, 0xB7, 0xBD,
0xC3, 0x5A, 0x3B, 0xC8, 0x69, 0x67,
    0xBD, 0xFB, 0xE8, 0x67, 0xBF, 0xCA, 0x9D, 0xE9, 0x74, 0x08, 0xE7, 0xCE,
0x77, 0x78, 0x06, 0x89, 0x32, 0x57,
    0xD6, 0xF1, 0xF1, 0x8F, 0x7D, 0xFE, 0x1F
};

const uint8_t spTHIR_[] PROGMEM = {
    0x04, 0xA8, 0xBE, 0x5C, 0x00, 0xDD, 0xA5, 0x11, 0xA0, 0xFA, 0x72, 0x02,
    0x74, 0x97, 0xC6, 0x01, 0x09, 0x9C, 0xA6, 0xAB, 0x30, 0x0D, 0xCE, 0x7A,
0xEA, 0x6A, 0x4A, 0x39, 0x35, 0xFB,
    0xAA, 0x8B, 0x1B, 0xC6, 0x76, 0xF7, 0xAB, 0x2E, 0x79, 0x19, 0xCA, 0xD5,
0xEF, 0xCA, 0x57, 0x08, 0x14, 0xA1,
    0xDC, 0x45, 0x64, 0x03, 0x00, 0xC0, 0xFF, 0x03
};
```

```
const uint8_t spFIF_[] PROGMEM = {
    0x08, 0x98, 0x31, 0x93, 0x02, 0x1C, 0xE0, 0x80, 0x07, 0x5A, 0xD6, 0x1C,
    0x6B, 0x78, 0x2E, 0xBD, 0xE5, 0x2D, 0x4F, 0xDD, 0xAD, 0xAB, 0xAA, 0x6D,
0xC9, 0x23, 0x02, 0x56, 0x4C,
    0x93, 0x00, 0x05, 0x10, 0x90, 0x89, 0x31, 0xFC, 0x3F
};

const uint8_t sp_TEEN[] PROGMEM = {
    0x09, 0x58, 0x2A, 0x25, 0x00, 0xCB, 0x9F, 0x95, 0x6C, 0x14, 0x21, 0x89,
    0xA9, 0x78, 0xB3, 0x5B, 0xEC, 0xBA, 0xB5, 0x23, 0x13, 0x46, 0x97, 0x99,
0x3E, 0xD6, 0xB9, 0x2E, 0x79, 0xC9,
    0x5B, 0xD8, 0x47, 0x41, 0x53, 0x1F, 0xC7, 0xE1, 0x9C, 0x85, 0x54, 0x22,
0xEC, 0xFA, 0xDB, 0xDD, 0x23, 0x93,
    0x49, 0xB8, 0xE6, 0x78, 0xFF, 0x3F
};

const uint8_t spTWENTY[] PROGMEM = {
    0x0A, 0xE8, 0x4A, 0xCD, 0x01, 0xDB, 0xB9, 0x33, 0xC0, 0xA6, 0x54, 0x0C,
    0xA4, 0x34, 0xD9, 0xF2, 0x0A, 0x6C, 0xBB, 0xB3, 0x53, 0x0E, 0x5D, 0xA6,
0x25, 0x9B, 0x6F, 0x75, 0xCA, 0x61,
    0x52, 0xDC, 0x74, 0x49, 0xA9, 0x8A, 0xC4, 0x76, 0x4D, 0xD7, 0xB1, 0x76,
0xC0, 0x55, 0xA6, 0x65, 0xD8, 0x26,
    0x99, 0x5C, 0x56, 0xAD, 0xB9, 0x25, 0x23, 0xD5, 0x7C, 0x32, 0x96, 0xE9,
0x9B, 0x20, 0x7D, 0xCB, 0x3C, 0xFA,
    0x55, 0xAE, 0x99, 0x1A, 0x30, 0xFC, 0x4B, 0x3C, 0xFF, 0x1F
};

const uint8_t spT[] PROGMEM = {
    0x01, 0xD8, 0xB6, 0xDD, 0x01, 0x2F, 0xF4, 0x38, 0x60, 0xD5, 0xD1, 0x91,
    0x4D, 0x97, 0x84, 0xE6, 0x4B, 0x4E, 0x36, 0xB2, 0x10, 0x67, 0xCD, 0x19,
0xD9, 0x2C, 0x01, 0x94, 0xF1,
    0x78, 0x66, 0x33, 0xEB, 0x79, 0xAF, 0x7B, 0x57, 0x87, 0x36, 0xAF, 0x52,
0x08, 0x9E, 0x6B, 0xEA, 0x5A,
    0xB7, 0x7A, 0x94, 0x73, 0x45, 0x47, 0xAC, 0x5A, 0x9C, 0xAF, 0xFF, 0x07
};

const uint8_t spHUNDRED[] PROGMEM = {
    0x04, 0xC8, 0x7E, 0x5C, 0x02, 0x0A, 0xA8, 0x62, 0x43, 0x03, 0xA7, 0xA8,
    0x62, 0x43, 0x4B, 0x97, 0xDC, 0xF2, 0x14, 0xC5, 0xA7, 0x9B, 0x7A, 0xD3,
0x95, 0x37, 0xC3, 0x1E, 0x16, 0x4A,
    0x66, 0x36, 0xF3, 0x5A, 0x89, 0x6E, 0xD4, 0x30, 0x55, 0xB5, 0x32, 0xB7,
0x31, 0xB5, 0xC1, 0x69, 0x2C, 0xE9,
    0xF7, 0xBC, 0x96, 0x12, 0x39, 0xD4, 0xB5, 0xFD, 0xDA, 0x9B, 0x0F, 0xD1,
0x90, 0xEE, 0xF5, 0xE4, 0x17, 0x02,
    0x45, 0x28, 0x77, 0x11, 0xD9, 0x40, 0x9E, 0x45, 0xDD, 0x2B, 0x33, 0x71,
0x7A, 0xBA, 0x0B, 0x13, 0x95, 0x2D,
    0xF9, 0xF9, 0x7F
};

const uint8_t spTHOUSAND[] PROGMEM = {
    0x0C, 0xE8, 0x2E, 0xD4, 0x02, 0x06, 0x98, 0xD2, 0x55, 0x03, 0x16, 0x68,
    0x7D, 0x17, 0xE9, 0x6E, 0xBC, 0x65, 0x8C, 0x45, 0x6D, 0xA6, 0xE9, 0x96,
0xDD, 0xDE, 0xF6, 0xB6, 0xB7, 0x5E,
    0x75, 0xD4, 0x93, 0xA5, 0x9C, 0x7B, 0x57, 0xB3, 0x6E, 0x7D, 0x12, 0x19,
0xAD, 0xDC, 0x29, 0x8D, 0x4F, 0x93,
    0xB4, 0x87, 0xD2, 0xB6, 0xFC, 0xDD, 0xAC, 0x22, 0x56, 0x02, 0x70, 0x18,
0xCA, 0x18, 0x26, 0xB5, 0x90, 0xD4,
```

```
    0xDE, 0x6B, 0x29, 0xDA, 0x2D, 0x25, 0x17, 0x8D, 0x79, 0x88, 0xD4, 0x48,
0x79, 0x5D, 0xF7, 0x74, 0x75, 0xA1,
    0x94, 0xA9, 0xD1, 0xF2, 0xED, 0x9E, 0xAA, 0x51, 0xA6, 0xD4, 0x9E, 0x7F,
0xED, 0x6F, 0xFE, 0x2B, 0xD1, 0xC7,
    0x3D, 0x89, 0xFA, 0xB7, 0x0D, 0x57, 0xD3, 0xB4, 0xF5, 0x37, 0x55, 0x37,
0x2E, 0xE6, 0xB2, 0xD7, 0x57, 0xFF,
    0x0F
};

const uint8_t spAND[] PROGMEM = {
    0xA9, 0x6B, 0x21, 0xB9, 0x22, 0x66, 0x9F, 0xAE, 0xC7, 0xE1, 0x70, 0x7B,
    0x72, 0xBB, 0x5B, 0xDF, 0xEA, 0x56, 0xBB, 0x5C, 0x65, 0xCB, 0x66, 0xC5,
0x3D, 0x67, 0xD7, 0xAB, 0x6D,
    0x2E, 0x64, 0x30, 0x93, 0xEE, 0xB1, 0xCD, 0x3D, 0x92, 0xB9, 0x9A, 0xDA,
0xB2, 0x8E, 0x40, 0x12, 0x9A,
    0x6A, 0xEB, 0x96, 0x8F, 0x78, 0x98, 0xB3, 0x2A, 0xB4, 0xD3, 0x48, 0xAA,
0x2F, 0x7D, 0xA7, 0x7B, 0xFB,
    0x0C, 0x73, 0x71, 0x5C, 0xCE, 0x6E, 0x5C, 0x52, 0x6C, 0x73, 0x79, 0x9A,
0x13, 0x4B, 0x89, 0x45, 0xE9,
    0x6E, 0x49, 0x42, 0xA9, 0x57, 0xFF, 0x3F
};

const uint8_t spMINUS[] PROGMEM = {
    0xE6, 0x28, 0xC4, 0xF8, 0x44, 0x9A, 0xFB, 0xCD, 0xAD, 0x8D, 0x2A, 0x4E,
    0x4A, 0xBC, 0xB8, 0x8C, 0xB9, 0x8A, 0xA9, 0x48, 0xED, 0x72, 0x87, 0xD3,
0x74, 0x3B, 0x1A, 0xA9, 0x9D, 0x6F,
    0xB3, 0xCA, 0x5E, 0x8C, 0xC3, 0x7B, 0xF2, 0xCE, 0x5A, 0x5E, 0x35, 0x66,
0x5A, 0x3A, 0xAE, 0x55, 0xEB, 0x9A,
    0x57, 0x75, 0xA9, 0x29, 0x6B, 0xEE, 0xB6, 0xD5, 0x4D, 0x37, 0xEF, 0xB5,
0x5D, 0xC5, 0x95, 0x84, 0xE5, 0xA6,
    0xFC, 0x30, 0xE0, 0x97, 0x0C, 0x0D, 0x58, 0x40, 0x03, 0x1C, 0xA0, 0xC0,
0xFF, 0x03
};

const uint8_t spMILLI[] PROGMEM = {
    0x6E, 0xF0, 0x8A, 0xB3, 0x4B, 0xEB, 0xC6, 0xAE, 0x36, 0xA7, 0x1A, 0x3A,
    0x54, 0x53, 0xD6, 0xDC, 0xEC, 0x66, 0x23, 0xDF, 0x58, 0x26, 0x43, 0xB4,
0xCD, 0xEA, 0x74, 0x5D, 0x94, 0x46,
    0xF0, 0x96, 0x3B, 0x9D, 0x79, 0x98, 0x26, 0x75, 0xDB, 0xB3, 0xD7, 0xB6,
0xF5, 0x90, 0xA8, 0x91, 0x9F, 0xEA,
    0x9E, 0xEE, 0xE9, 0x9B, 0x20, 0x7D, 0xCB, 0xFF, 0x03
};

const uint8_t spVOLTS[] PROGMEM = {
    0xA0, 0xDA, 0xA2, 0xB2, 0x3A, 0x44, 0x55, 0x9C, 0xFA, 0xB0, 0xBA, 0x46,
    0x72, 0xDA, 0xD1, 0xDB, 0xAE, 0x47, 0x59, 0x61, 0xED, 0x28, 0x79, 0xED,
0x45, 0xAF, 0x5A, 0xDF, 0x60, 0xF4,
    0x39, 0x69, 0xAB, 0x63, 0xD9, 0x3B, 0xD2, 0xBC, 0x24, 0xA5, 0xF5, 0xB6,
0x0F, 0x80, 0x01, 0x3E, 0x63, 0x65,
    0xC0, 0x5F, 0x63, 0x12, 0x90, 0x80, 0x06, 0x24, 0x20, 0x01, 0x0E, 0xFC,
0x3F
};

/* Say any number between -999,999 and 999,999 */
void sayNumber (long n) {

    if (n < 0) {
```

```
    voice.say (spMINUS);
    sayNumber (-n);
} else if (n == 0) {
    voice.say (spZERO);
} else {

    if (n >= 1000) {
        int thousands = n / 1000;
        sayNumber (thousands);
        voice.say (spTHOUSAND);
        n %= 1000;

        if ((n > 0) && (n < 100))
            voice.say (spAND);
    }

    if (n >= 100) {
        int hundreds = n / 100;
        sayNumber (hundreds);
        voice.say (spHUNDRED);
        n %= 100;

        if (n > 0)
            voice.say (spAND);
    }

    if (n > 19) {
        int tens = n / 10;

        switch (tens) {
        case 2:
            voice.say (spTWENTY);
            break;

        case 3:
            voice.say (spTHIR_);
            voice.say (spT);
            break;

        case 4:
            voice.say (spFOUR);
            voice.say (spT);
            break;

        case 5:
            voice.say (spFIF_);
            voice.say (spT);
            break;

        case 6:
            voice.say (spSIX);
            voice.say (spT);
            break;

        case 7:
            voice.say (spSEVEN);
            voice.say (spT);
```

```
        break;

    case 8:
        voice.say (spEIGHT);
        voice.say (spT);
        break;

    case 9:
        voice.say (spNINE);
        voice.say (spT);
        break;

    }

    n %= 10;
}

//number less than or equal to 19
switch (n) {
case 1:
    voice.say (spONE);
    break;

case 2:
    voice.say (spTWO);
    break;

case 3:
    voice.say (spTHREE);
    break;

case 4:
    voice.say (spFOUR);
    break;

case 5:
    voice.say (spFIVE);
    break;

case 6:
    voice.say (spSIX);
    break;

case 7:
    voice.say (spSEVEN);
    break;

case 8:
    voice.say (spEIGHT);
    break;

case 9:
    voice.say (spNINE);
    break;

case 10:
    voice.say (spTEN);
```

```
            break;

        case 11:
            voice.say (spELEVEN);
            break;

        case 12:
            voice.say (spTWELVE);
            break;

        case 13:
            voice.say (spTHIR_);
            voice.say (sp_TEEN);
            break;

        case 14:
            voice.say (spFOUR);
            voice.say (sp_TEEN);
            break;

        case 15:
            voice.say (spFIF_);
            voice.say (sp_TEEN);
            break;

        case 16:
            voice.say (spSIX);
            voice.say (sp_TEEN);
            break;

        case 17:
            voice.say (spSEVEN);
            voice.say (sp_TEEN);
            break;

        case 18:
            voice.say (spEIGHT);
            voice.say (sp_TEEN);
            break;

        case 19:
            voice.say (spNINE);
            voice.say (sp_TEEN);
            break;
        }
    }
}

const int phototransistor = A0;
int value = 0;

void setup () {
    pinMode (phototransistor, INPUT);
}

void loop () {
    value= analogRead(0) * 5.000 / 1.023;
```

```
    sayNumber (value);
    voice.say(spMILLI);
    voice.say(spVOLTS);
    delay (2000);
}
```

Learning Objective 3 Projects

Using parts from both the RC Rover and Snapino kits, create a programmable robot.

Programmed RC rover

Parts Needed

Qty	Part
1	10x7 Base grid
1	Rover Body
1	Snapino module with connection cable
1	(S1) Slide Switch
1	(U8) Motor Control IC
4	2-Snap wire
8	Jumper Wires
1	3-snap to pin wire

First Layer

1. Snapino module 5V on C8 , GND on C6, and D11 on F6
2. Motor Control IC on C3-C5-F3-F5, with positive side of unit on B6 and B8
3. (S1) Slide Switch on G2-G4

Second Layer

4. 2-Snap wire F4 G4
5. 2-Snap wire D5 D6
6. 2-Snap wire E5 E6
7. 2-Snap wire F5 F6
8. Snap jumper wire C4C6
9. Snap jumper wire C5C7
10. Snap jumper wire C3 to top left snap on back of Rover
11. Snap jumper wire D3 to bottom left snap on back of Rover
12. Snap jumper wire E3 to top middle snap on back of Rover
13. Snap jumper wire F3 to bottom middle snap on back of Rover
14. Snap jumper wire G2 to top right snap on back of Rover

Third Layer

15. Snap side of Snap to pin wire on G4 other end to VIN on header of Snapino
16. Snap jumper wire C4 to bottom right snap on back of Rover

Code

```
//car go the square bot
const int leftForwardPin =  A0;
const int leftBackPin =  9;
const int rightForwardPin =  10;
const int rightBackPin =  11;

//stops all motors and delays for given time in milliseconds
void stop(int time) {
    digitalWrite(leftForwardPin, LOW);
    digitalWrite(leftBackPin, LOW);
    digitalWrite(rightForwardPin, LOW);
    digitalWrite(rightBackPin, LOW);
    delay (time);
}

//forward function the stop is used to stop all motors
void forward(int time) {
    digitalWrite(leftForwardPin, HIGH);
    digitalWrite(leftBackPin, LOW);
    digitalWrite(rightForwardPin, HIGH);
    digitalWrite(rightBackPin, LOW);
    delay(time);
    stop(0);
}

//a Right turn function the stop is used to stop all motors
void right(int time) {
    digitalWrite(leftBackPin, HIGH);
    digitalWrite(rightForwardPin, HIGH);
    delay(time);
    stop(0);
}

void setup() {
    // put your setup code here, to run once:
    pinMode(leftForwardPin, OUTPUT);
    pinMode(leftBackPin, OUTPUT);
    pinMode(rightForwardPin, OUTPUT);
    pinMode(rightBackPin, OUTPUT);
    stop(0);
}

void loop() {
    // put your main code here, to run repeatedly:
    forward(1500);
    stop(3000);
    right(800);
    stop(3000);
}
```

Bonus Project

Talking braille keyboard

Parts needed

Qty	Part
1	10 by 7 Snap Circuit Bread board grid
1	(U31) Snapino Module
1	USB Standard A male to Standard B
1	speaker
3	2 Snap Jumper wire
7	snap to male pin Wire Jumper
7	(S2) Push button
4	2-snap wire
2	3-snap wire
1	4-snap wire
1	6-snap wire

First Layer

1. Snapino U31, A1 - D3, with 5V on D1
2. Snap side of a Snap to male pin Wire Jumper A6 with pin side to the left will connect to U31 in Layer two
3. Snap side of a Snap to male pin Wire Jumper B6 with pin side to the left will connect to U31 in Layer two
4. Snap side of a Snap to male pin Wire Jumper C6 with pin side to the left will connect to U31 in Layer two
5. Snap side of a Snap to male pin Wire Jumper D6 with pin side to the left will connect to U31 in Layer two
6. Snap side of a Snap to male pin Wire Jumper E6 with pin side to the left will connect to U31 in Layer two
7. Snap side of a Snap to male pin Wire Jumper F6 with pin side to the left will connect to U31 in Layer two
8. Snap side of a Snap to male pin Wire Jumper G6 with pin side to the left will connect to U31 in Layer two
9. 6-snap wire, A8- F8
10. 2 connector wire, C9 - D9
11. Switch S2, C10 - E10
12. 3-snap wire, G8 - G10

Second Layer

13. 2 Snap Jumper D3 -G9
14. Snap to male Pin Wire Jumper on G6, Pin to D0 in header of U31
15. Snap to male Pin wire on F6, Pin to D1 in header of U31
16. Snap to male Pin Wire Jumper on e6, Pin to D4 in header of U31
17. Snap to male Pin Wire Jumper on D6, Pin to D8 in header of U31
18. Snap to male Pin Wire Jumper on C6, Pin to D5 in header of U31
19. Snap to male Pin Wire Jumper on B6, Pin to D6 in header of U31
20. Snap to male Pin Wire Jumper on A6, Pin to D7 in header of U31
21. Switch S2, A6 - A8
22. Switch S2, B6 - B8
23. Switch S2, C6 - C8
24. 4 Connector snap wire, D6 - D8

25. Switch S2, E6 - E8
26. Switch S2, F6 - F8
27. Switch S2, G6 - G8
28. 2-snap wire, C9 - C10
29. 3-snap wire, E10 - G10

Third Layer

30. 2-snap wire, F8 - G8

Sound Output for Braille Keyboard

The Speaker out has been left out in the above instructions. There is room for it on the board after you build the above braille keyboard. Depending on which piece you want to use, the instructions are a bit different. Each of the different possibilities has been listed in the next sections. If you are just using the Speaker and two Snap jumper wires in the parts list, using the speaker section will finish the build. If you want the sound to be louder and you have one of the Snap Circuit power amps U then follow the "Using the power amp and speaker" section. Finally, if you want to use a head set or speaker, follow the "Using the 3.5 MM JA" section to add links to each section above.

Using the Snap Circuit Speaker

31. Place speaker G1 - G3

32. Place 2 Snap Jumper D3 - G3

33. Place 2 Snap Jumper A3 G

Using Speaker with power Amp U4

Extra Parts Needed

Quantity	Part
1	(U4) Power Amp
2	2-snap jumper wire
2	2-snap wire
1	3-snap wire

Orientation and description of U4

Place part flat on table with side of the part that has least snaps toward you. From left to right, clockwise around the body of the part.

Bottom left snap Amplified output signal

Top Left Negative

Middle top Input unamplified signal

Top right Filtered power (Not used)

Bottom right Positive

First Layer

34. 3power Amp U4 E1 - F3 Positive snap at E1
35. Speaker G1 - G3

Second Layer

36. 2 Snapconnector D1 - E1
37. 2-Snap wire F3 - G3
38. 2 Snap Jumper wire A3 -F2
39. 2 Snap jumper wire E3 - G1

Third layer

40. 3-Snap wire D3 F3

Using the 3.5 MM jack part (JA)

Extra Parts Needed

Quantity	Part
1	Snap Circuit 3.5 jack part JA
1	2 snap jumper wire

Orientation and description of JA

The Snap Circuit 3.5 audio Jack part JA is a 45, 45, 90-degree triangle that takes up 3 Snaps on the long side, and the 90 degree angle of the right triangle snap is one row or column from the two snaps on the long side in the middle of them both.

Place the 45, 45, 90-degree triangle with hypotenuse parallel to the table edge and the 3.5 jack facing you. The Snap at the 90-degree angle is the right channel snap. The snap to the left of the 3.5 jack is the Left channel snap. The snap to the right of the 3.5 jack is the common ground. We will not use the common ground in this build so that the sound will be in both ears of a pair of earphones.

First Layer

41. 3.5 Jack (JA) G1 - G3 with 90-degree on F2

Second Layer

42. Snap jumper wire A3 -F2
43. Snap Jumper wire G1 other side placed in Layer 3

Third Layer

44. Snap jumper wire D3

Code

```
#include
#include &amp;quot;Vocab_US_Large.h&amp;quot;
#include &amp;quot;Vocab_US_TI99.h&amp;quot;

Talkie voice;

const byte INPUT_PINS[7]= {0,1,4,5,6,7,8};

#define DOT_1  0x01
#define DOT_2  0x02
#define DOT_3  0x04
#define DOT_4  0x08
#define DOT_5  0x10
#define DOT_6  0x20
#define SPACE   0x40

uint8_t buttonsDown =0;
uint8_t lastButtonsDown =0;

void setup()
{
    for (int i=0; i&amp;lt;7; i++)
        pinMode(INPUT_PINS[i], INPUT_PULLUP);

    voice.say(sp4_POWER);
    delay (100);
    voice.say(sp4_ON);
    delay (500);
    voice.say(sp2_CIRCUIT);
    delay (100);
    voice.say(sp4_READY);
}

void loop()
{
    uint8_t  *letter=0;
    byte db=0;

    while (db&amp;lt;2) {
        for (byte i=0; i&amp;lt;7; i++)
            buttonsDown |=  ((digitalRead(INPUT_PINS[i])==LOW)?(1&amp;lt;
```