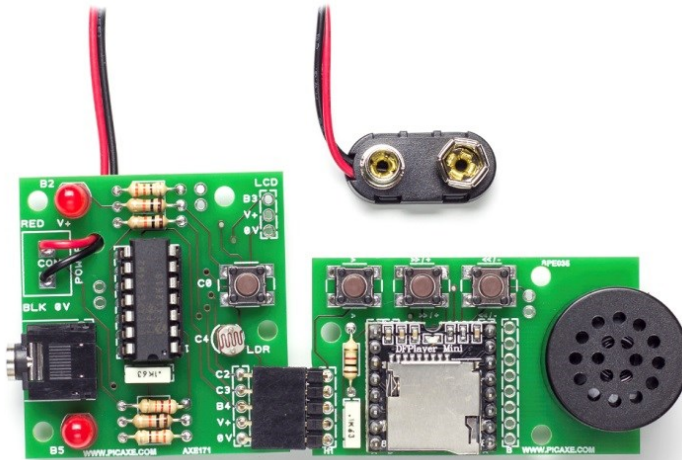


AXE171 PICAXE-14 Audio Kit



The AXE171 is a small PICAXE-14M2 project designed to connect to the SPE035 MP3 player module. Both kits are provided within the PICAXE-14 Audio kit.

The SPE035 Serial MP3 player consists of a small MP3 player module mounted on a PCB with serial connector, test switches and 8 ohm speaker. It provides a simple and low cost way to add MP3 tune playback to any PICAXE project.

The MP3 audio files (music, speech etc.) are copied onto a microSD card (not included) which is then inserted into the MP3 player. A simple 3 wire connection to the PICAXE project then allows playback and control of the audio tunes.

AXE171 Features:

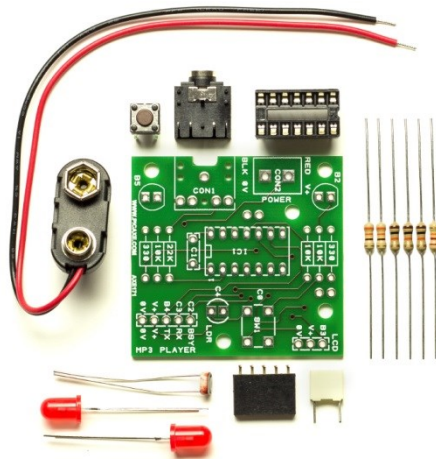
1. PICAXE-14M2 Microcontroller
2. Two LED outputs and two spare outputs e.g. for Serial LCD or Servo
3. Push switch and LDR light sensor
4. MP3 player interface

SPE035 Features:

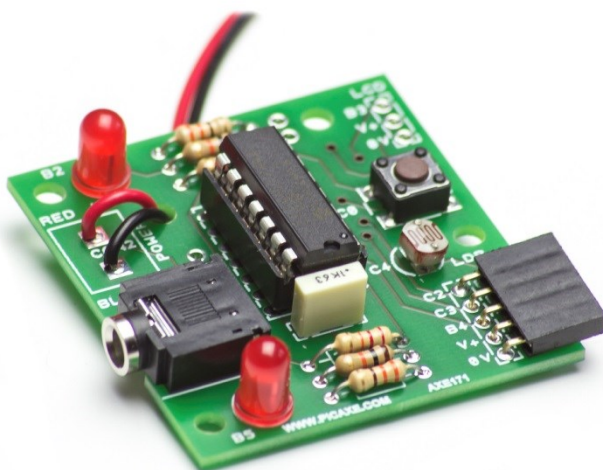
1. Supports MP3 files with all most common file sampling frequencies (kHz):
 - i. 8 / 11.025 / 12 / 16 / 22.05 / 24 / 32 / 44.1 / 48
2. High quality playback on 8 ohm speaker (included) with 90dB dynamic range
3. PCB pads for alternate stereo line out audio connection
4. Supported file format: MP3 / WAV
5. Supports 1GB to 32GB microSD card (FAT16 or FAT32)
6. 30 volume settings (1= mute, 30 = full volume)
7. 6 equalizer settings (0=Normal, 1=Pop, 2=Rock, 3=Jazz, 4=Classic, 5=Bass)
8. TTL serial control playback mode, at 9600 baud rate (PICAXE serout at T9600_8)
9. Power supply can be 3.3 to 5.2V DC
10. On board switches for playback testing

For the full datasheet see www.picaxe.com/docs/axe171.pdf

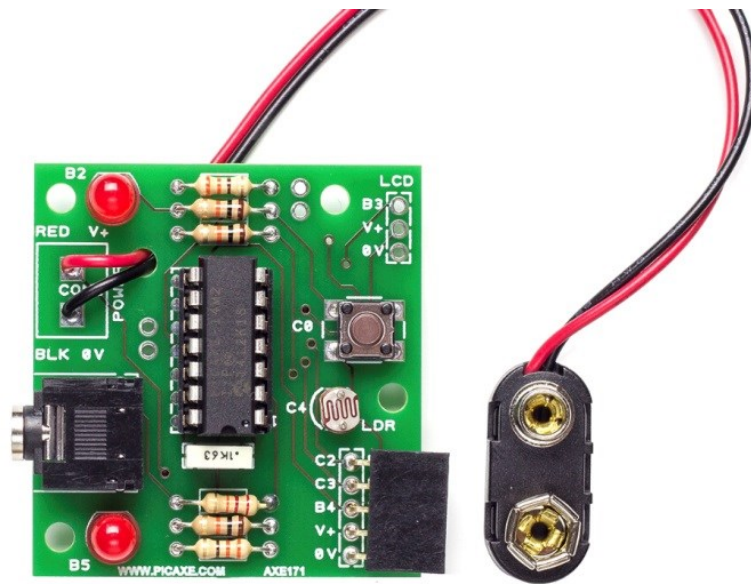
1.1 AXE171 Kit Contents



1	PCB		AXE171 pcb
1	SW1	SEN030	6mm push switch
2	R1,6	RES-330	330 resistor (orange orange brown gold)
3	R2,3,5	RES-10K	10k resistor (brown black orange gold)
1	R4	RES-22K	22k resistor (red red orange gold)
1	C1	CAP001	100nF capacitor
1	IC1	ICH014	14 pin IC socket
1	IC1	AXE017M2	PICAXE-14M2 (pin 1 placed next to C1)
1	LDR	SEN002	LDR
2	L1-2	LED001	5mm red LED (long leg +)
1	H1	CON043	5 pin r/a socket
1	CON1	CON039	3.5mm download socket
1	CON2	BAT016	4.5V battery clip



1.2 AXE171 Assembly



Note – Before assembly decide whether you may prefer to solder the LEDs, LDR, switch (and also the SPE035 speaker) on the reverse side of the PCB (therefore soldering these component pins on the top side of the board). This is often useful when mounting the PCB through a cardboard model. The PCB supplied is a high quality plated through PCB with solder pads on both sides.

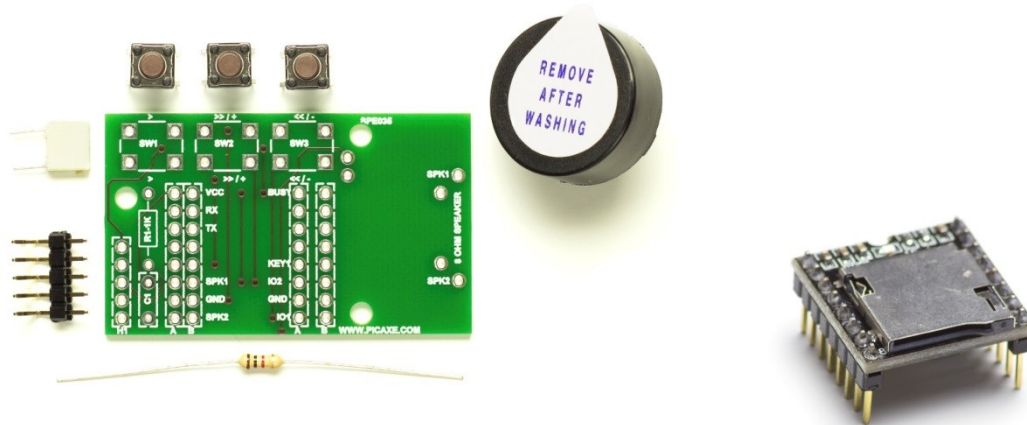
1. Solder the resistors in position R1-R6 and the capacitor in position C1.
2. Solder the switch in position SW1
3. Solder the 5 pin right angle socket (H1) and the 3.5mm download socket (CON1) in position. Make sure this download socket 'snaps' flat on the PCB.
4. Solder the 14 pin IC socket in position, noting the pin 1 notch is to the bottom of the board.
5. Solder the LEDs and LDR to the board. You may wish to leave these with long legs or to solder them on the reverse of the PCB. Make sure the LEDs are correctly aligned with the marking on the PCB, long leg through the hole marked +.
6. Solder the battery clip to the board. You may choose to thread the two wires through the large hole first to make a more durable connection (see photo above). Make sure the red (RED) and black (BLK) wires are the correct way around.

Insert the PICAXE-14M2 into the socket. Note that **pin 1 (notched end of chip)** is placed next to the capacitor at the **bottom** of the PCB so that the chip is 'upside down' (see photo above).

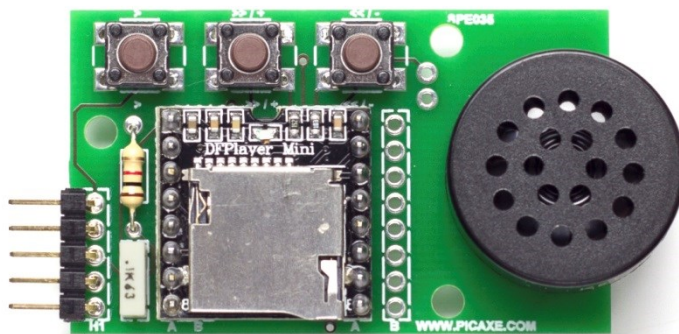
The board is designed to be used with a 4.5V (3xAAA) battery pack. Do **NOT** use a 9V PP3 battery.

The additional holes and pads on the PCB are not required at this stage and so can be ignored.

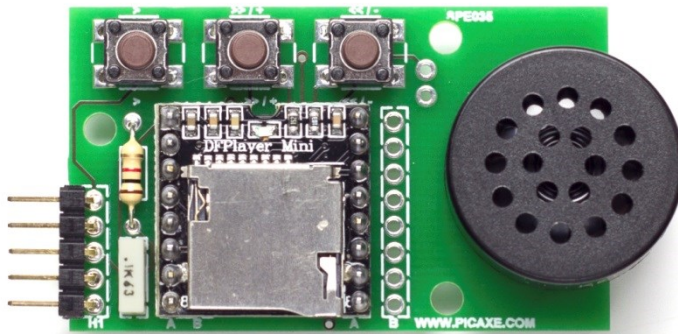
2.0 SPE035 Kit Contents



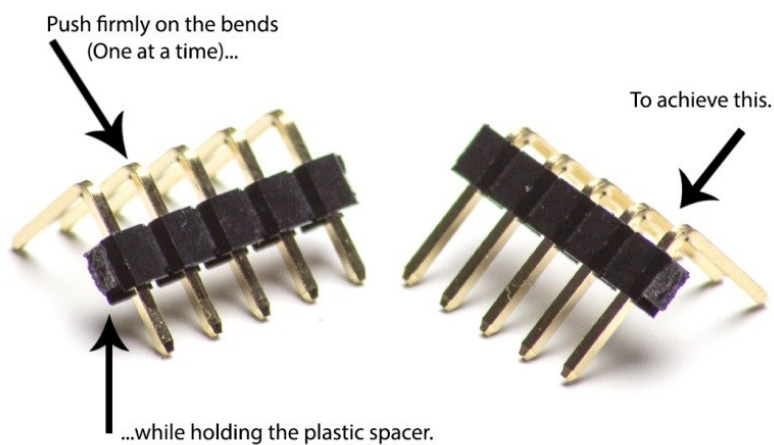
1	PCB		SPE035 pcb
3	SW1-3	SEN030	6mm push switch
1	R1	RES-1K	1k resistor (brown black red gold)
1	C1	CAP001	100nF capacitor
1	H1	CON042	5 pin r/a header
1	SPK1	SPE015	8 ohm PCB mount speaker
2	MP3 socket	CON050	8 way 2.54mm sockets (optional)
1	MP3 player	SPE033	DFPlayer Mini (or similar clone)



2.1 SPE035 Assembly



- Solder the 1k resistor in position R1 and capacitor in position C1.
- Solder the three switches in positions SW1 to SW3
- Solder the MP3 module in the appropriate position A or B (see note 1.1 above). The module may be soldered directly onto the board or the optional 8 way connectors supplied may be used (if you wish to be able to remove the module from the PCB in the future).



- On the 5 way connector carefully push hard on the corner of the first pin (e.g. with a coin) so that the short end slides through the plastic holder, therefore making the short end longer. Do this for all 5 pins. Place the longer end through the PCB in position H1 and solder in position. The connector is adjusted and used like this so it lies closer to the top of the PCB.
- Solder the speaker wires into positions SPK1 and SPK2. There are 3 different solder pad positions to allow for different speaker sizes and shapes. If you wish to use a different size of speaker it should be an '8 ohm' type. The speaker can be connected either way round.

3.0 Loading MP3 files onto your micro SD card (not included)

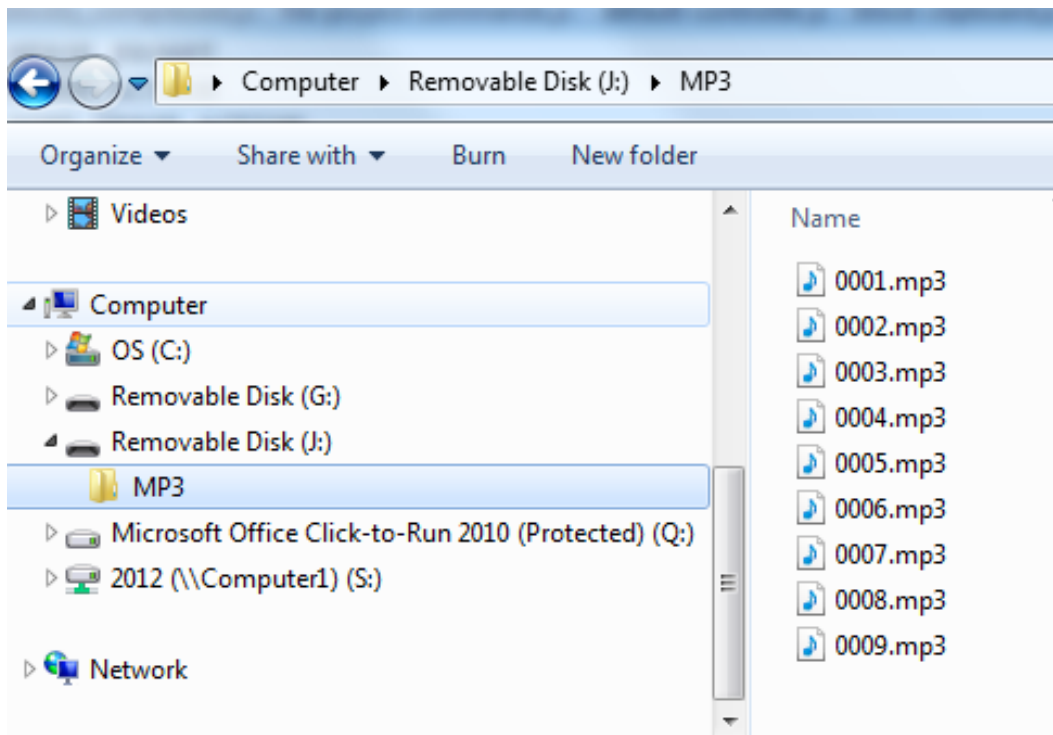
IMPORTANT – Note the microSD card socket on the top of the MP3 module is a ‘push to eject’ style socket (so you must press the card inwards slightly to eject it). If you simply pull hard on the card without ejecting it you may cause physical damage to the socket and/or module!

MP3 music/speech files may be copied onto the card using a mobile phone, tablet or computer (a separate SD card adapter may be required to insert the microSD card into a computer). MicroSD cards from 1GB to 32GB are supported. Many people may already have a surplus micro SD card from an old mobile phone which will be ideal.

The music filenames **must** have filenames starting with 0001.mp3, 0002.mp3, 0003.mp3 etc. and be saved in a subfolder called \MP3 on the microSD card.

A set of sample MP3 files for testing may be downloaded from

www.picaxe.com/downloads/mp3.zip



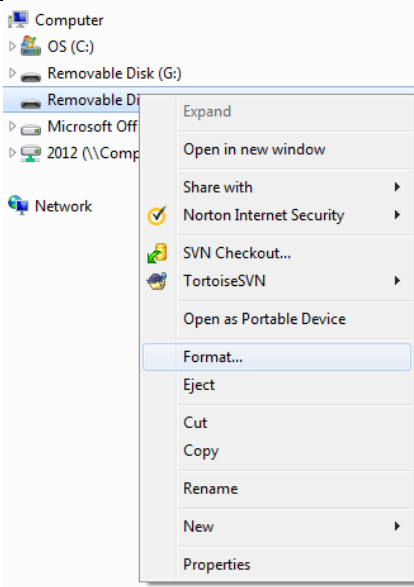
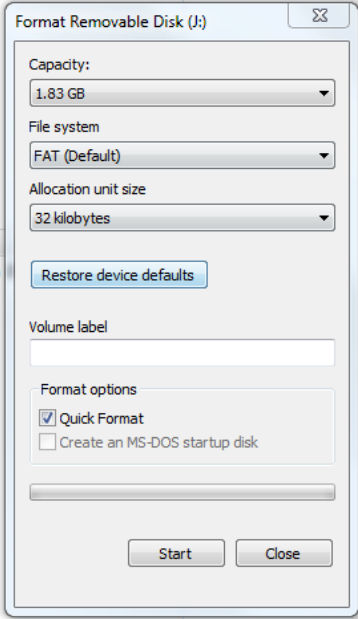
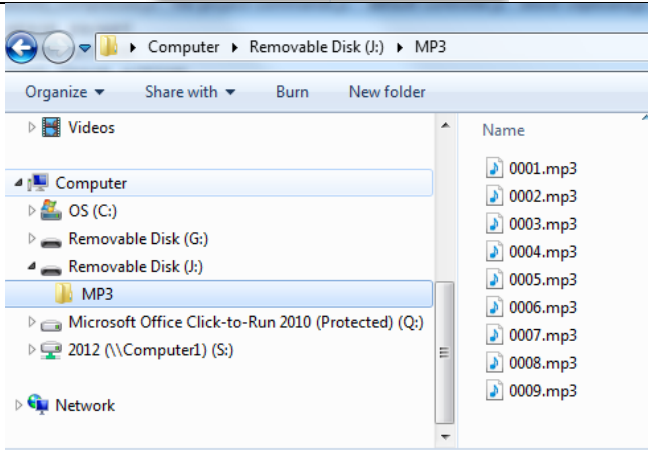
So the MP3/WAV files copied onto the microSD card should:

- Use a filename that starts with 4 numeric characters e.g. 0001.mp3, 0002.wav
- Use a filename that starts with numbers between 0001.mp3 and 0255.mp3
- End with .mp3 or .wav
- Be saved within a folder called \MP3 on the microSD card
- Be unprotected files - DRM ‘copyright protected’ music files are not supported

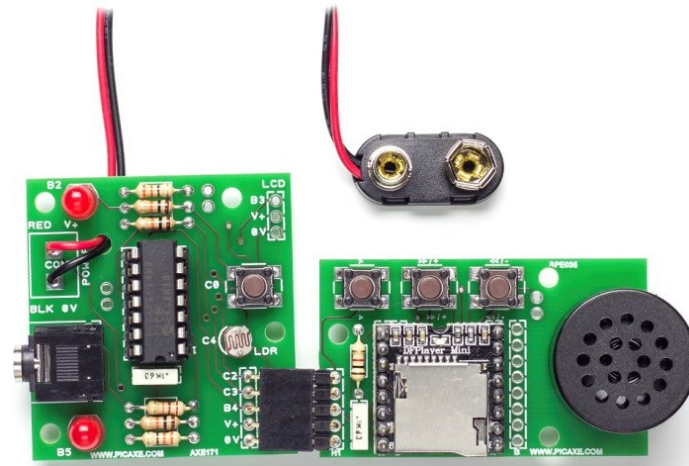
Once the music and speech files have been copied onto the micro SD card carefully and gently insert the card into the socket on the MP3 module.

3.1 Step by step guide to copying MP3 files on to the SD card:

If using an old recycled microSD card it is best to 'Quick Format' it before use.

<p>Step 1.</p> <p>Insert the microSD card into the computer (an adapter may be required), right click over the microSD card drive and select 'Format...'</p>	
<p>Step 2.</p> <p>Make sure 'Quick Format' is selected.</p> <p>Double check that it is definitely the microSD card drive that is selected (as formatting permanently deletes all data)</p> <p>Click Start</p>	
<p>Step 3.</p> <p>Once formatting is complete create a new folder called MP3 on the microSD card</p> <p>Step 4.</p> <p>Copy the MP3 files into the \MP3 sub folder. It is best to copy the files one at a time in order (see notes in 2.0)</p>	

3.2 Testing the MP3 files (using the on-board switches).

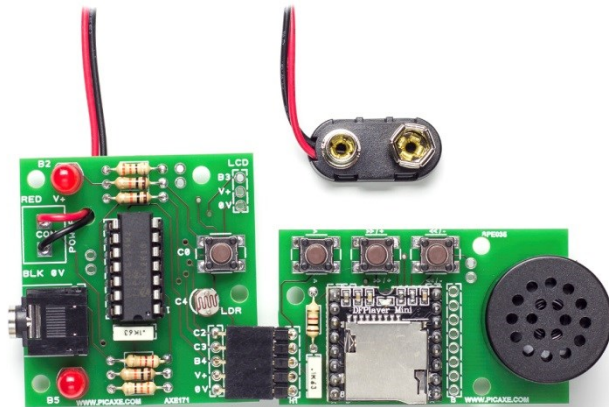


1. Make sure the microSD card containing MP3 files has been inserted into the MP3 player.
2. Connect power (4.5V battery box) to the AXE171 battery clip.
3. Press the left play (>) switch, the first tune found will then play.
4. A short press of the centre (>> / +) switch will move to the next track.
5. A short press of the right (<< / -) switch will move to the previous track.
6. A long press of the centre (>> / +) switch will increase the volume.
7. A long press of the right (<< / -) switch will decrease the volume.

*Note that, as with most MP3 players, the next/previous track push buttons operate on the FAT file system sort, not the alphabetical filename sort. So the next/previous track played will be according to the order that the files were originally saved onto the card, **not** via an alphabetical sort of the filenames currently on the card. Use a search engine search as Google to find a 'FAT filename sorter' utility program if the push switch playback order is essential and must be amended.*

When playback is controlled by serial commands from a PICAXE chip the files are called directly by filename number, so the sort order on the microSD card is then not important.

4.0 Example PICAXE programs



PICAXE-14M2 Input / Output Connections

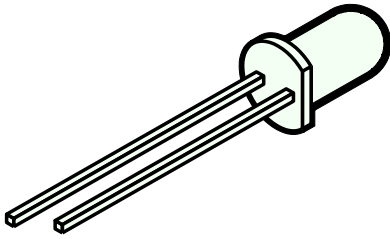
B.0	Programming
B.1	Spare - Serial LCD or Servo (optional)
B.2	Top LED
B.3	Spare - Serial LCD or Servo (optional)
B.4	TX to MP3 player
B.5	Bottom LED
C.0	Push switch
C.1	Spare
C.2	BUSY from MP3 player
C.3	RX from MP3 player
C.4	LDR

Examples are provided for Blockly, Flowchart and BASIC programming options.

To view the included sample files within 'PICAXE Editor 6' use the File>Open Samples menu and select 'AXE171 – PICAXE 14 Audio Kit'

4.1 Switching the LEDs on and off

4.1.0 What is an LED?



A Light Emitting Diode (LED) is an electronic component that gives out light when current passes through it. An LED is a special type of diode. A diode is a component that only allows current to flow in one direction.

Therefore when using a diode, it must always be connected the correct way around.

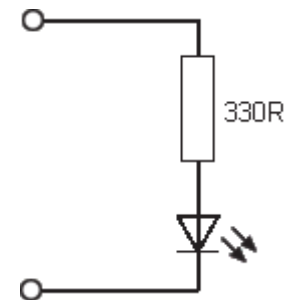
The positive (anode) leg of an LED is longer than the negative (cathode) leg (shown by the bar on the symbol). The negative leg also has a flat edge on the plastic casing of the LED.

4.1.1 What are LEDs used for?

LEDs are mainly used as indicator lights. Red and green LEDs are commonly used on electronic appliances like televisions to show if they are switched on or in 'standby' mode. LEDs are available in many different colours, including red, yellow, green and blue. Special 'ultrabright' LEDs are used in safety warning devices such as the 'flashing lights' used on bicycles. Infra-red LEDs produce infra-red light that cannot be seen by the human eye but can be used in devices such as video remote-controls.

4.1.2 Using LEDs.

LEDs only require a small amount of current to work, which makes them much more efficient than bulbs (this means, for instance, that if powered by batteries the LEDs will light for a much longer time than a bulb would). If too much current is passed through an LED it will be damaged, and so LEDs are normally used together with a 'series' resistor that protects the LED from too much current.

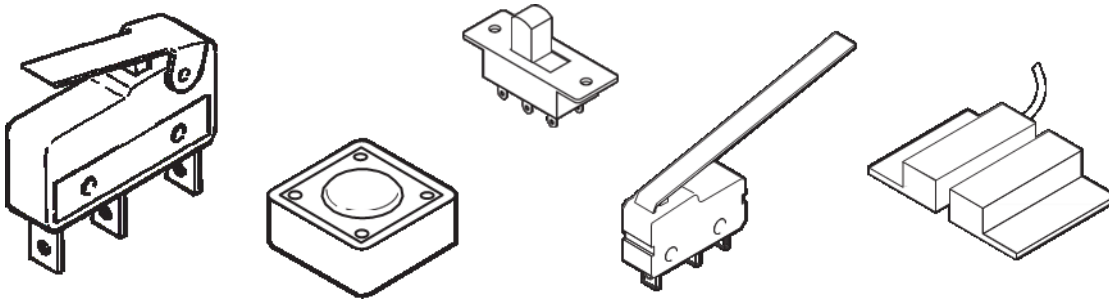


The value of the resistor required depends on the battery voltage used. For a 4.5V battery pack a 330R resistor can be used, and for a 3V battery pack a 120R resistor is appropriate.

4.1.3 Connecting the LED to a PICAXE microcontroller

Because the LED only requires a small amount of current to operate, it can be directly connected between the microcontroller output pin and 0V (with the series protection resistor).

4.2 Using the push switch



4.2.0 What are switches?

A 'switch' is a type of digital sensor, a sensor that can only be 'on' or 'off'. Switches detect movement.

There are a large number of different types of switches e.g.:

- push switches that detect a momentary 'push'
- micro-switches with long levers that detect small movements tilt-switches that detect jolting
- reed-switches that detect a magnet being moved

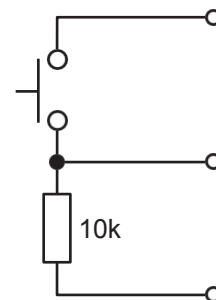
4.2.1 What are switches used for?

Push switches are commonly used on device like keypads. Micro-switches are used in burglar alarms to detect if the cover is removed from the alarm box. Reed switches are used to detect doors and windows being opened and tilt switches are often used to detect movement in devices such as toys, hair-dryers and tool-box alarms.

4.2.2 Using switches with a PICAXE microcontroller

A switch is often used with a resistor as shown in the diagram. The value of the resistor is not that important, but a 10k resistor is often used. When the switch is 'open' the 10k resistor connects the microcontroller input pin down to 0V, which gives an 'off' (logic level 0) 0V signal to the microcontroller input pin.

When the switch is activated, the input pin is connected to the positive battery supply (V+). This provides an 'on' (logic level 1) signal to the microcontroller.



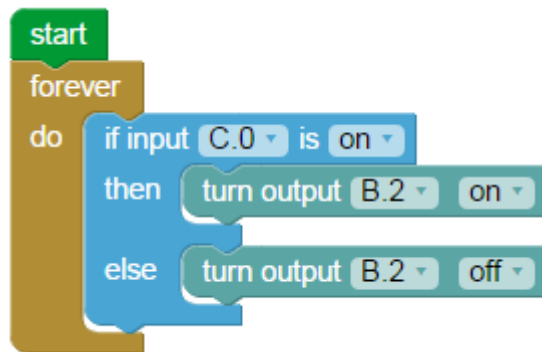
4.2.3 Testing the switch

The AXE171 project board has a switch on: Input pinC.0

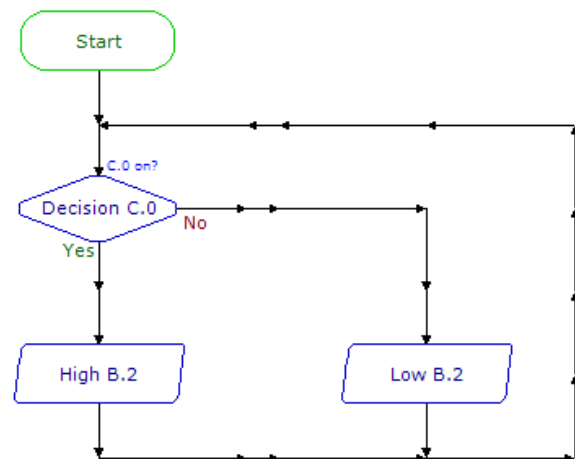
Therefore make sure the correct pin number is selected within each PICAXE command.

After connecting the switch it can be tested by a simple program like this. This program will switch an output on and off according to whether the switch is pushed or not.

Blockly



Flowchart



BASIC

```

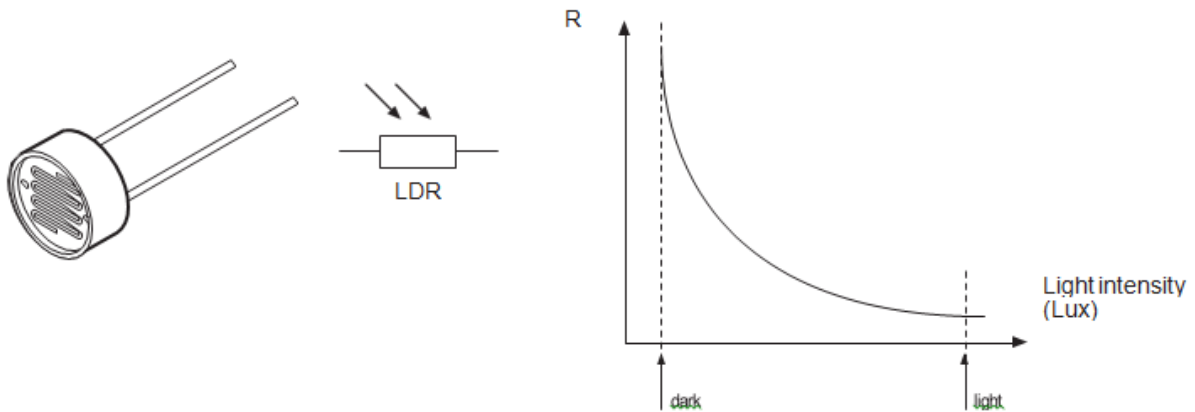
do
  if pinC.0 = 1 then      ; test pinC.0
    high B.2             ; LED on
  else
    low B.2              ; LED off
  end if
loop

```

4.3 Using the LDR

4.3.0 What is an LDR?

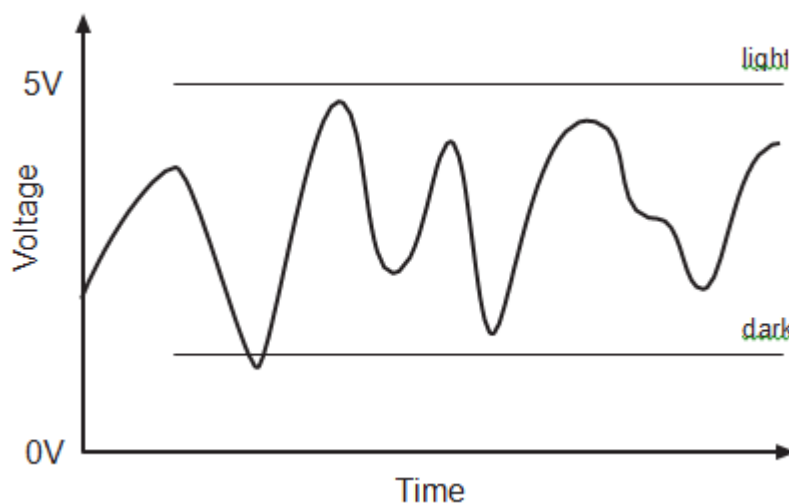
A Light Dependent Resistor (LDR) is special type of resistor that reacts to changes in light level. The resistance of the LDR changes as different amounts of light fall on the top 'window' of the device. This allows electronic circuits to measure changes in light level.



4.3.1 What are LDRs used for?

LDRs are used in automatic street lamps to switch them on at night and off during the day. They are also used within many alarm and toys to measure light levels.

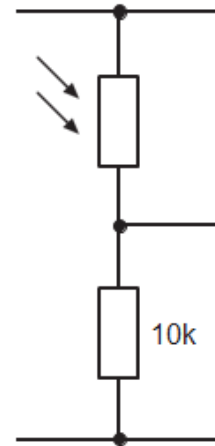
The LDR is a type of analogue sensor. An analogue sensor measures a continuous signal such as light, temperature or position (rather than a digital on-off signal like a switch). The analogue sensor provides a varying voltage signal. This voltage signal can be represented by a number in the range 0 and 255 (e.g. very dark = 0, bright light = 255).



4.3.2 Using an LDR with a PICAXE microcontroller.

The electronic circuit for using the LDR is known as a 'potential divider'. A suitable value for the lower resistor is 10k. The voltage at the centre position of the potential divider varies as the resistance of the LDR changes in different light conditions.

The microcontroller contains an 'analogue to digital converter (ADC)' which converts the varying voltage signal into a number between 0 and 255. This number can then be stored in a variable and compared against a desired 'threshold point'.



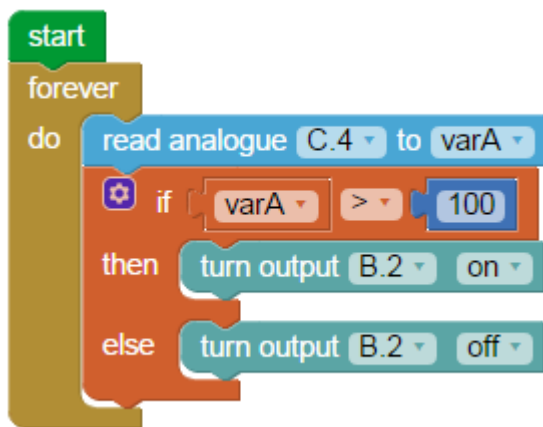
4.2.3 Testing the LDR

The AXE171 project board has an LDR on: Input C.4

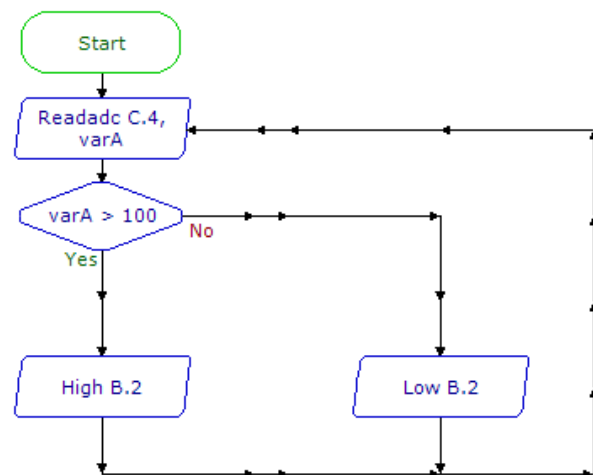
Therefore make sure the correct pin number is selected within each PICAXE command.

This program will switch an output on and off according to whether the light level is greater than 100. But who do you know 100 is the correct threshold value to use? See the next section...

Blockly



Flowchart



BASIC

```

symbol varA = b1
do
  readadc C.4, varA      ; read ADC value on C.4
  if varA > 100 then    ; test value
    high B.2           ; LED on
  else
    low B.2            ; LED off
  end if
loop
    
```

4.2.3 Calibrating the analogue sensor

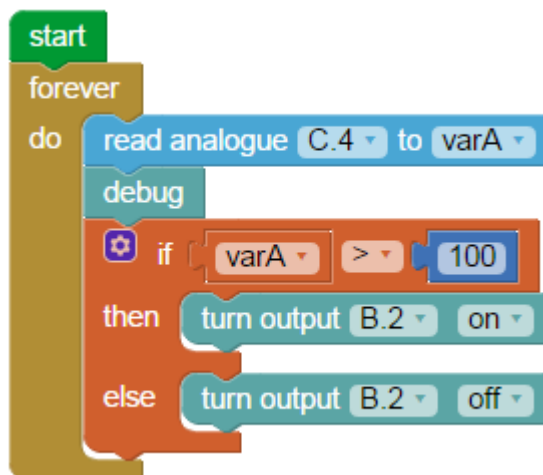
When using an analogue sensor you may not always know what the correct threshold value to use. The threshold value is normally a number that is half way between the lowest (darkest) and highest (brightest) analogue values possible.

Fortunately the PICAXE system has a special 'debug' command which can transmit the value of the variable to the computer screen. Using this command you can therefore run an experiment – for instance using your hand to cover the LDR to calculate the highest and lowest possible values.

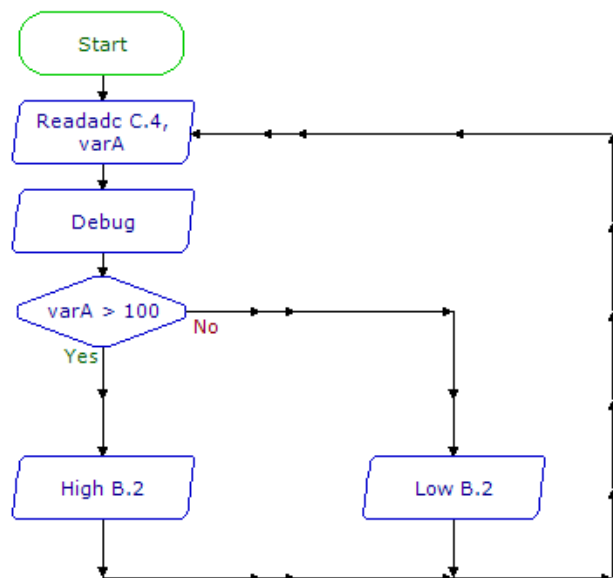
The threshold value to use is the value that is half way between these maximum and minimum values.

To try this out modify the previous program by inserting an extra 'debug' command (found in the Serial Commands section) after the readadc command.

Blockly



Flowchart



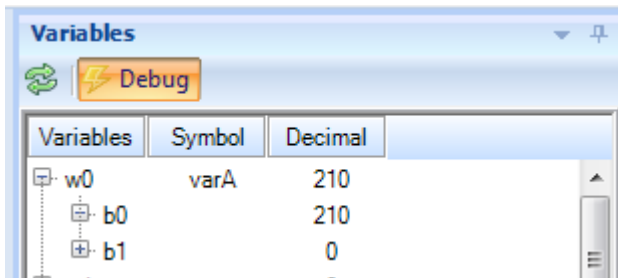
BASIC

```

symbol varA = b1
do
  readadc C.4, varA      ; read ADC value on C.4
  debug                  ; send value to computer
  if varA > 100 then     ; test value
    high B.2             ; LED on
  else
    low B.2              ; LED off
  end if
loop
  
```


After downloading this new program open the Debug panel on the computer screen. The Debug panel will then show the value of the light reading.

In **PICAXE Editor 6** the debug panel is found in the Code Explorer on the right hand side of the screen.



In the **Blockly Chrome App** the Debug panel can be opened from the PICAXE menu.

Blocks	PICAXE BASIC			Debug X	Javascript	XML
b0	211	\$D3	%11010011	w0	00211	\$00D3
b1	000	\$00	%00000000			
b2	000	\$00	%00000000	w1	00000	\$0000
b3	000	\$00	%00000000			

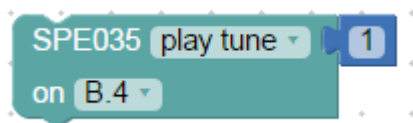
Once the experiment is complete select a new threshold value half way between the maximum and minimum values found. Then modify the original program to use this value rather than the value 100 that was originally used.

5.0 Playing MP3 Music Files using SPE035

Note that the MP3 module requires up to 4 seconds (pause 4000) at power on to read the tune filename data from the microSD card. Depending on the quantity of MP3 files this can take up to 4 seconds, but may be much quicker. Therefore do not send any commands within this initial period.

5.1 Using SPE035 in Blockly

Blockly v1.0.5 (and later) has a special SPE035 command within the Outputs menu. This simplifies use of all the most common SPE035 commands. Simply drag out the block and select the desired command from the drop down list.

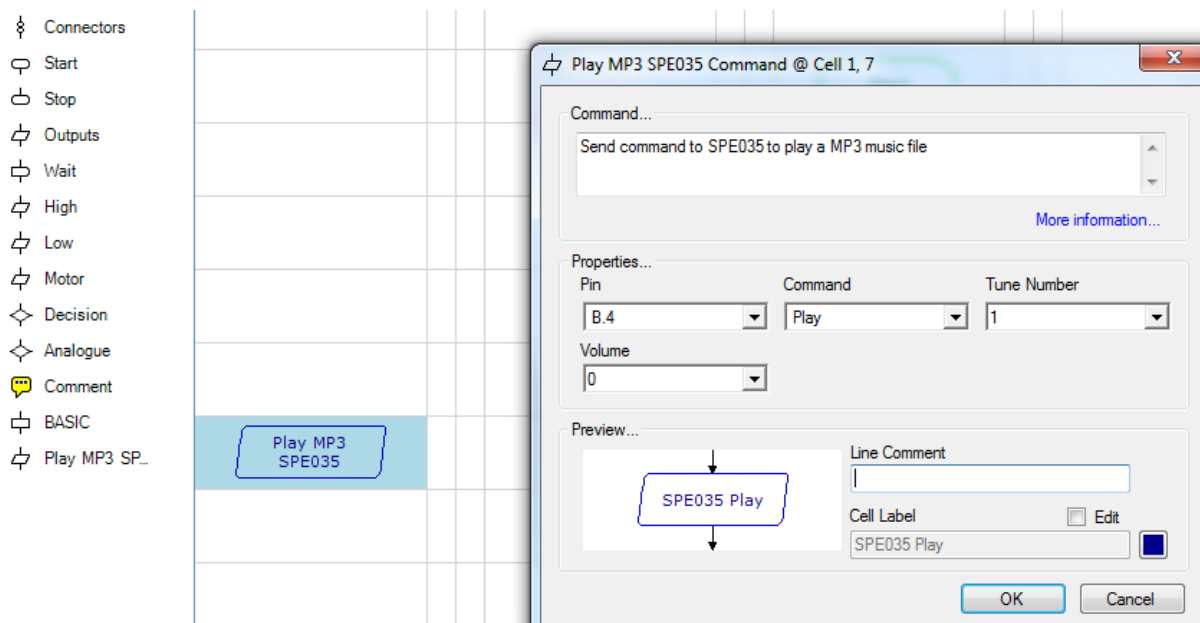


A number (or variable) block only needs to be attached to the SPE035 block for the 'play tune' and 'set volume' commands.

5.2 Using SPE035 in Flowcharts

PE6 v6.0.8.4 (and later) has a special SPE035 flowchart command. To add this command right click over any whitespace area in the Flowchart Toolbox and select 'Add New Item' > 'Play MP3 SPE035'

This special flowchart block simplifies use of all the most common SPE035 commands. Simply drag out the block, double click on it to edit and select the desired command from the drop down list.



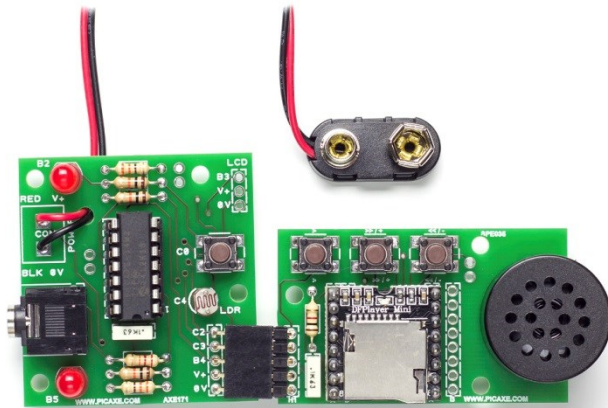
The 'tune number' value is only used when the 'Play' command is selected.

The 'volume' value is only used when the 'Set Volume' command is selected.

5.3 Simple 'Play Tune' Example

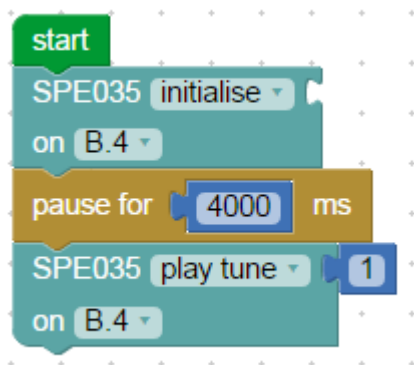
The AXE171 project board connects to SPE035 by:
 Output B.4
 Input pinC.2

Therefore make sure the correct pin number is selected within each PICAXE command.

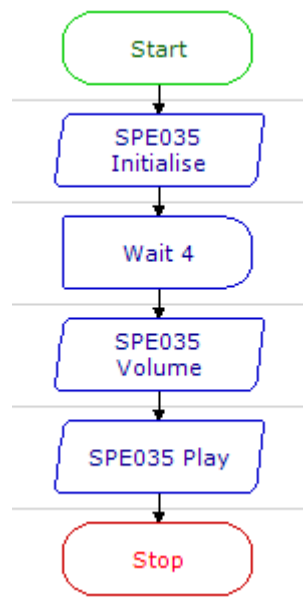


To use the SPE035 it is first necessary to issue the 'initialise' command to the module (this tells the module to read files from the microSD card). Depending on the quantity/size of the MP3 files this can take up to 4 seconds, but may be much quicker. The play command is then used to start tune number 1 (0001.mp3).

Blockly



Flowchart



BASIC

```
Symbol TX           = B.4
```

```
Symbol RX           = C.3
```

```
Symbol BUSY_PIN     = pinC.2
```

```
Symbol BAUD_FREQ    = M8
```

```
Symbol BAUD         = T9600_8
```

```
Symbol cmd          = b0
```

```
Symbol arg          = w1 ; b3:b2
```

```
Symbol arg.lsb     = b2
```

```
Symbol arg.msb     = b3
```

```
High TX           ; set TX pin high for idle high serial
```

```
Pause 2000
```

```
SerTxd("Starting", CR, LF )
```

```
SerTxd("Select microSD Card", CR, LF )
```

```
cmd = $09 : arg = $0002 : Gosub Send
```

```
Pause 4000
```

```
SerTxd("Play MP3 folder song 0001.mp3", CR, LF )
```

```
cmd = $12 : arg = 0001 : Gosub Send
```

```
Pause 1000
```

```
Stop
```

```
Send:
```

```
SetFreq BAUD_FREQ
```

```
Pause 10
```

```
SerOut TX, BAUD, ( $7E, $FF, $06, cmd, $00, arg.msb, arg.lsb, $EF )
```

```
SetFreq MDEFAULT
```

```
Return
```

5.4 Play Tune when Light Level Drops

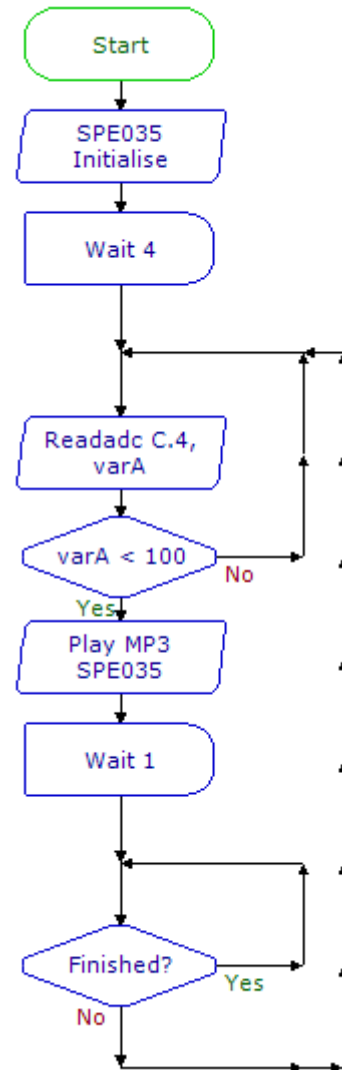
This example shows how to play a tune when the light level drops below 100.

The end of the previous tune is detected by checking whether the MP3 BUSY signal (connected to PICAXE input C.2) is high. The BUSY signal is low (off) when the tune is playing and high (on) at all other times. Therefore the program waits for input pinC.2 to go high (on) before restarting.

Blockly



Flowchart



BASIC

```

Symbol TX          = B.4

Symbol RX          = C.3
Symbol BUSY_PIN   = pinC.2

Symbol BAUD_FREQ  = M8
Symbol BAUD       = T9600_8

Symbol cmd        = b0

Symbol arg        = w1 ; b3:b2
Symbol arg.lsb   = b2
Symbol arg.msb   = b3

Symbol varA      = w2

High TX          ; set TX pin high for idle high serial

Pause 2000
SerTxd("Starting", CR, LF )

SerTxd("Select microSD Card", CR, LF )
cmd = $09 : arg = $0002 : Gosub Send
Pause 4000

Do
  ReadADC C.4, varA
  If varA < 100 then
    SerTxd("Play MP3 folder song 0001", CR, LF )
    cmd = $12 : arg = 1 : Gosub Send
    Pause 1000
    Do While BUSY_PIN = 0
      Pause 100
    Loop
  End If
Loop

Send:
  SetFreq BAUD_FREQ
  Pause 10
  SerOut TX, BAUD, ( $7E, $FF, $06, cmd, $00, arg.msb, arg.lsb, $EF )
  SetFreq MDEFAULT
  Return

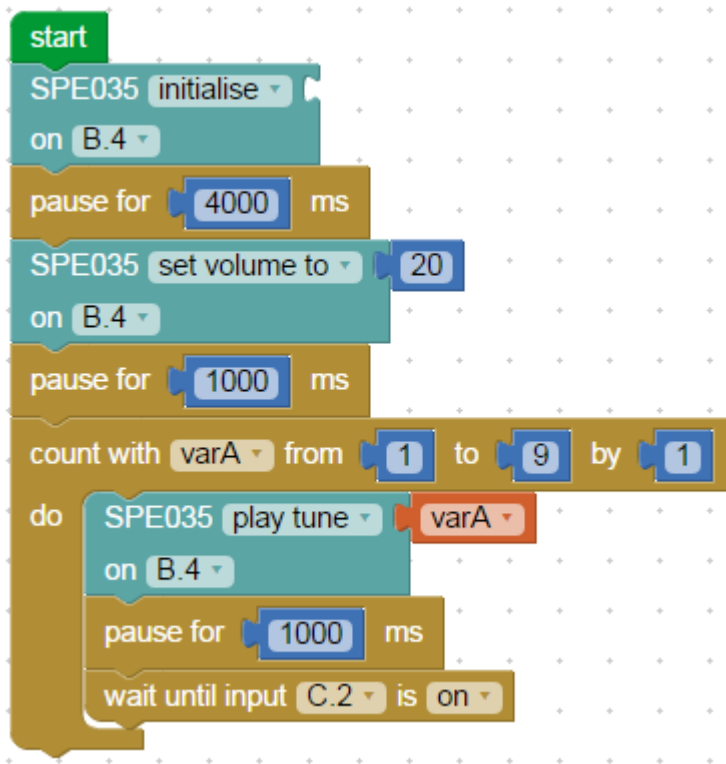
```

5.5 Play Multiple Tunes using a Variable

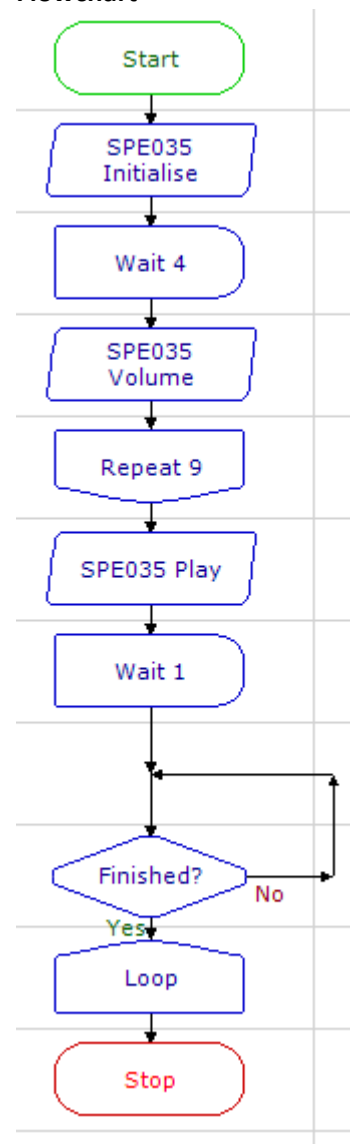
This example shows how to set the volume (valid values are 1 to 30) and then use a variable (varA) to play the tunes 0001.mp3 to 0009.mp3 in turn.

The end of the previous tune is detected by checking whether the MP3 BUSY signal (connected to PICAXE input C.2) is high. The BUSY signal is low (off) when the tune is playing and high (on) at all other times. Therefore the program waits for input pinC.2 to go high (on) before playing the next tune.

Blockly



Flowchart



BASIC

```

Symbol TX          = B.4

Symbol RX          = C.3
Symbol BUSY_PIN    = pinC.2

Symbol BAUD_FREQ   = M8
Symbol BAUD        = T9600_8

Symbol cmd         = b0

Symbol arg         = w1 ; b3:b2
Symbol arg.lsb     = b2
Symbol arg.msb     = b3

Symbol varA        = w2

High TX           ; set TX pin high for idle high serial

Pause 2000
SerTxd("Starting", CR, LF )

SerTxd("Select microSD Card", CR, LF )
cmd = $09 : arg = $0002 : Gosub Send
Pause 4000

SerTxd("Set volume 20", CR, LF )
cmd = $06 : arg = 20 : Gosub Send
Pause 1000

For varA = 1 To 9
  SerTxd("Play MP3 folder song 000", #varA, CR, LF )
  cmd = $12 : arg = varA : Gosub Send
  Pause 1000
  Do While BUSY_PIN = 0
    Pause 100
  Loop
Next

Sertxd("Done", CR, LF )
Stop

Send:
  SetFreq BAUD_FREQ
  Pause 10
  SerOut TX, BAUD, ( $7E, $FF, $06, cmd, $00, arg.msb, arg.lsb, $EF )
  SetFreq MDEFAULT
  Return

```


6.0 Circuit Diagram

To be added.