Rover 6 Mechatronics project

K Barnsdale.
January 2020

Table of Contents

1 Introduction

1.1 House rules

- Contribution money required next week
- Cleanup and chairs up always before anyone leaves
- Safety glasses to be used when soldering and wash hands after soldering
- Burning plastic = exclusion from class

1.2 Introduction to Mechatronics

- What is Mechatronics ?
- Mechatronics is combining Electronics, Mechanics and Software
- Career opportunities

1.3 Distribute bags or boxes, and instructions

- Names on storage bags
- Handout instructions so they can show parents
- Distribute template file and final programming file to everyone, and make it 'read only'

1.4 Rover6 intro

Introduction to Rover6 and its micro controller

1.5 Software:

1.5.1 New introductions

- Introduce the Picaxe development environment on screen
- Introduce template file (see instructions document)
- Always 'save as' new work file to separate folder
- Introduce 'comments'
- Introduce help documents
- Introduce commands 'High' , 'Low', and multiple nouns (e.g. High Red, Left)
- Introduce the 'Do/ Loop'.
- Introduce the 'Pause'.
- Introduce the on screen simulator

1.5.2 Tasks

- Using High, low, pause. Turn an output ON for half a second, OFF for 1 second, ON for half a second, then OFF.

Sample code for above - flash LED.bas

```
high red        ;turn on LED
pause 500       ;pause for half second
low red         ;turn off LED
pause 1000      ;pause for a second
high red        ;turn on LED
pause 500       ;pause for half second
low red         ;turn off LED
```

- Note - we can include two outputs (green and red) in the same command (`low green, red`) saving space. Later we will use this to turn motors and LEDs on and off together.

Sample code for above - Cycle LEDS.bas

```
do
      high red          ;turn on red  LED
      pause 1000        ;pause for 1 second
      high green        ;turn on green LED
      pause 500         ;pause for half a second
      low green, red    ;turn off both
      pause 200         ;pause for 200 milli seconds
loop        ;go back and do it again
```

1.6 Extension work

- Simulate a traffic light, with timing of three colours
- Sequence the LED outputs with one LED on, shifting along the line
- Find ‘pause’ and ‘do’ commands in help documents

2 Build and test PCB

2.1 Hardware:

Start building PCB, using instructions

1. Soldering rules
2. Discover the parts in the storage cabinet, and components drawers.
3. Component identification
4. Component polarity
5. Only one component at a time
6. Soldering instruction and practice
7. All components as low down on board as possible
8. Explain USB and Light sensor fitting details

2.2 Test PCB hardware

Thorough visual check by teacher (Joints, bridges, polarity etc)

2.3 Instructions on first programming

- Use instructions document for guidance
    - Plug in USB cable to check red light on USB module.
    - Setup 'COM port' number (not AXE027) (see instructions document)
    - Click 'Check Picaxe type connected' to confirm connection.
    - Setup 'PICAXE type' to be PICAXE 18M2 (see instructions document)
- Review downloading hang ups and errors. See Rescue Remedies in file 'Rover 6 Picaxe Rescue remedies'
- Use the simple test program 'PCB check.bas' in *instructions document* to test finished PCB

2.4 Software (if time)

2.4.1 Revise (pre requisites)

- PICAXE editor screen layout
- Use of Template
- 'save as' new program file to separate folder
- Comments
- High , Low, and multiple nouns (e.g. High Red, Left)
- Do/ Loop
- Pause

2.4.2 New introductions

- 'Pulsout' command

2.4.3 Tasks

- Flash an LED or three, or copy/paste PCB test code from instructions.
- Try different timing
- Try simulating traffic lights
- Try sequence red yellow green yellow red etc

Sample code - sequence LEDs.bas

```
do
      high red          ;turn ON red  LED
      pause 100         ;pause for 100 milli second
      low red           ;turn OFF red LED
      high yellow       ;turn on yellow LED
      pause 100
      low yellow
      high green        ;turn on green LED
      pause 100
      low green
loop              ;go back and do it again
```

- Using Pulsout creates a pulse on the output pin. The number is the pulse time in 10's of microseconds. (You can subtract two zeros to get the number of milliseconds)

Sample code –flash LEDs using 'pulsout' - pulsout flash.bas

```
do
      pulsout red, 10000    ;flash red for 100 milli sec
      pulsout yellow, 10000 ;flash yellow for 100 milli sec
      pulsout green, 10000  ;flash green for 100 milli sec
      pulsout yellow, 10000
loop              ;go back and do it again
```

2.5 Extension (if time)

- What is the shortest LED pulse your human eye can see?
- More soldering practice, make a structure, or animal, from resistors

3 Start writing software on Rover6 - PCB only

3.1 Software

3.1.1 Revise (pre requisites)

- Revise downloading process, and what to look for on downloading screen
- Revise downloading hang ups and errors.
- Revise use of template, and 'save as' file to separate folder
- Revise high, low, pause, do/loop
- Revise Comments

3.1.2 New introductions

- If / else/ endif
- Commands – 'play' and 'sound'
- Command 'sleep'

3.1.3 Tasks

- If / else/ endif. We test inputs (bumper switches, light sensor) with the IF statement. A bumper switch input =1 when pressed, a light sensor input=1 when light. The program lines after 'then' will run if the condition is true. The program lines after 'else' will run if the condition is not true. Every 'IF' statement must end with and 'endif' statement.

- Using if/endif– turn an LED on using the left bumper switch

Sample code - bumper LED.bas

```
do
      low red           ; LED off.
      if lbumper=1 then ; If left bumper pressed...
            high red    ;Red LED on
      endif
loop
```

- Turn red LED on only while right bumper is pressed

Sample code - bumper on off LED.bas

```
do
      If rbumper=1 then ; test if bumper is pressed (=1)
            high red          ; red LED on if pressed
      else
            low red           ; red LED off if not pressed
      endif
loop
```

- Try exchanging 'rbumper' input for light sensor input. Turn red LED on when light is detected

Sample code - light LED.bas

```
do
      If light=1 then   ; test if light is seen (=1)
            high red          ; red LED on if pressed
      else
            low red           ; red LED off if not pressed
      endif
loop
```

- Turn red LED on/off when each bumper is pressed

Sample code - L and R bumper LED.bas

```
do
      If rbumper=1 then
            high red    ;turn on red LED when right bumper pressed
      endif
      If lbumper=1 then
            low red     ;turn off red LED when left bumper pressed
      endif
loop
```

- The **'Play**' command

Create a 'Play' command with 'PLAY Spkr, song' , where 'song' is a number 0,1,2,3

Sample code - play command.bas

```
do
      play spkr,1 ;can choose song 0,1,2,3
loop
```

- The **'Sound '** command

SOUND Spkr, (note,duration,note,duration...) where 'note' is a number between 90 (low) and 120 (high), 'duration' is time in 10msec steps (100=1sec).
Experiment with different notes and note lengths.
Try sound command, two notes, each one second long

Sample code - sound.bas

```
do
      sound spkr,(90,100,120,100) ;the '100' means notes of one second each
loop
```

A reaction GAME with If/Else and Sound. Read the comments to find out how it works.

Sample code - reaction.bas

```
 do
      high yellow ; turn on LED….
      pause 3000  ; …. and wait for 3sec
      low yellow  ; as soon as LED turns off – push the bumper switch!
      pause 300    ; give them time to push the button
      if rbumper = 0 then     ;if not pushed by now, you are too slow
            sound spkr, (120,50,90,50)    ; loser!
            else
            sound spkr, (90,50,120,50)    ; winner!
      endif
loop
```

3.1.3.1 Sleep

We can put the Rover microcomputer into a deep sleep mode which takes almost no power from the battery. The number in the sleep command is the sleep time in steps of 2.3 seconds

Sample code – using sleep command – sleep.bas

```
do
      pulsout yellow, 1000  ;flash an LED at the start
      sleep 10 ; 10 x 2.3 seconds = 23 seconds
      play spkr,2
loop
```

3.1.4 Extension work

- Add LEDs to the winner / looser sections of reaction game.
- Make two different sounds on left/right bumper button presses
- Use light sensor to change the sound
- Flash LED in morse code

4 Build chassis, add motors, battery etc

4.1 Hardware

- Build chassis and motors, mount PCB and wiring. See instructions sheet

4.2 Software (if time)

4.2.1 Revise (pre requisites)

- Use of Template, 'save as' file to separate folder
- Do/loop
- High, low, Pause

4.2.2 New introductions

- High Left, High right (forward commands for motors)
- High lback, High rback (backward commands for motors)

4.2.3 Tasks

- Test motors and battery (forward, back, left right). Use test program (motor check.bas) in *instructions document*, or right your own using highs, lows and pauses, as below.

Sample code to test all motors - motor test.bas

```
do
      high right,left ;both motors on forward
      pause 2000  ;drive for 2 seconds
      low left    ;turn off forward left motor, turn to left
      pause 2000
      high lback  ;reverse left motor, spin
      pause 2000
      low right   ;turn off forward right motor, backwards turn
      pause 2000
      high rback  ;reverse right motor, drive backwards
      pause 2000
      low rback, lback ;turn off all reverse
loop
```

- Practice turns, spin. Calibrate timing for turns.
- Games (if time)
    1. Race to the line.
    2. Race around a square

4.3 Extension work

Try stopping with

a/ Both forward and back outputs 'Low'

b/ Both forward and back outputs 'High'

and see what difference there is in the way it stops.

(Outputs 'low' applies a brake to the wheel, outputs 'high' allows free wheel)

Sample code – different stopping techniques – way to stop.bas

```
      high right,left ;both motors on forward
      pause 1000  ;drive for 1 second
      low right, left ; brake!
      pause 1000

      high right,left ;both motors on forward
      pause 1000  ;drive for 1 second
      high lback, rback ; stop!
```

5 Driving and race skills

5.1 Software

5.1.1 Revise (pre requisites)

- Use of template, 'save as' file to separate folder
- Do/loop
- High, low, Pause

5.1.2 New introductions

NA

5.1.3 Tasks

- Calibrate speed/distance
- Race to over the line and return by reverse
- Calibrate turns and /or spin (how long should one wheel be stopped?)
- Race to line and turn around to return
- Rally to line (accurate timing)
- Use PWM to control speed

Sample code - to drive ahead and reverse – note it runs once - NO 'do / loop' here! race.bas

```
      high left, right, red, green ; all ON
      pause 5000              ;drive for a bit. You work out how long
      low left, right, red, green ; all forwards OFF
      high lback,rback, yellow      ; all reverse
      pause 5000              ;drive for a bit. You work out how long
      low lback, rback, yellow      ; all OFF
```

Sample code to do a turn and a spin – turn and spin.bas

```
do
      ; turn by stopping one wheel
      high left, right, red, green
      pause 2000        ; drive for a bit
      low left, red     ; turn off left motor
      pause 1550
      high left, red
      pause 2000        ; forward again for 2sec
            ; turn by spin
      low left, red     ; turn off forward drive
      high lback, yellow      ; turn on backward left
      pause 1000              ; spin left for 1 sec
      low lback, yellow             ; turn OFF backward left
      high left, red          ; turn on forward left
loop
```

5.2 Extension work

Drive a square figure of eight

5.3 Extra features

- Use buttons to control PWM up and down

Sample - This adjusts the speed by pressing the bumpers – PWM test.bas

```
pwmout left, 50, 100    ;setup PWM duty ratio
pwmout right, 50, 100
do
      pwmduty left,b0   ;change PWM ratio
      pwmduty right,b0
      low red
      if lbumper=1 then ;test for button press
            b0=b0+1
            high red
            ;pause 100 debug  ;optional readout of PWM duty b0
      endif
loop
```

6 Bumper driving

6.1 Software

6.1.1 Revise (pre requisites)

- Revise comments, template discipline
- Revise Do/loop, High, Low, Pause
- Revise If/Else/Endif

6.1.2 New introductions

- Introduce FOR/NEXT loop
- IF/endif to detect bumper activation and reverse/turn away
- For /Next loop to bumper driving to add 'rescue code' when Rover gets stuck

6.1.3 Tasks

Bumper driving

Using the bumper switches, use IF/ENDIF to detect when one has been hit, reverse and turn away. Note the use of LEDs with the motors to show what is going on. The working of this program can then be 'seen' with just the USB power connected. Note the reversals do not shut off the reversing motor, that is done when it gets back to the start of the loop.

Sample code for bumper driving, bumper drive.bas

```
do
      low rback,lback,red,green ;turn all off
      high right, left , yellow ;drive forward

      if lbumper=1 then ; test bumper
            low left, right ;stop!
            high rback, green ; reverse opposite wheel
            pause 500   ;do this for a while
      endif

      if rbumper=1 then ; test bumper
            low left, right ;stop!
            high lback, red ; reverse opposite wheel
            pause 500   ;do this for a while
      endif
loop
```

FOR/NEXT loop

FOR/NEXT loop used in a simple case. This goes around a loop five times and flashes an LED each time. Try simulating it to see counter (b0) change. Don't put this inside a do/loop, otherwise you will never see it finish!

Sample code for For/Next loop. This flashes red LED five times – for next.bas

```
      for b0=1 to 5 ;do the next bit five times
            high red    ;flash an LED
            pause 100
            low red
            pause 100
      next b0           ;do it again if not finished
```

6.2 Extra features

Incorporating the FOR/NEXT function into the bumper driving program.
Sometimes the Rover will get stuck against a wall. A simple 'reverse and turn' after a set time will rescue it. The main loop is run 2000 times (for/next loop) before it executes a reversal.

Sample code for bumper driving, bumper drive 2.bas

```
do
      for w0=1 to 2000  ;do this loop 2000 times
            low rback,lback,red,green ;turn all off
            high right, left , yellow ;drive forward

            if lbumper=1 then ; test bumper
                  low left, right ;stop!
                  high rback, green ; reverse opposite wheel
                  pause 500   ;do this for a while
            endif

            if rbumper=1 then ; test bumper
                  low left, right ;stop!
                  high lback, red ; reverse opposite wheel
                  pause 500   ;do this for a while
            endif
      next w0
; the next bit rescues the Rover if it gets stuck
      low left, right, yellow; intermittent direction change
      high lback;
      pause 500   ; short reversal
loop
```

6.3 Extension

- In the FOR/NEXT example, flash a green light when the it has finished, then put the whole thing in a DO/LOOP
- In a FOR/NEXT loop, try counting down.
- In the Bumper driving, make the rescue code only run when it hasn't seen a bumper activation for a while.

7 Multi tasking (optional if time allows)

7.1 Software

7.1.1 Revise (pre requisites)

- Revise comments, template discipline
- Revise Do/loop, High, Low, Pause
- Revise If/Else/Endif
- Pulsout, which uses a time in 10 microsecond units (eg 5000 = 50mSec)

7.1.2 New introductions

Introduce multitasking – (start1: start2: etc)

7.1.3 Tasks

Flash three LEDs with multi tasking.
Watching the flashes, ask your friend to guess the fastest/slowest?

Sample code to flash three LEDs with multi tasking- multitask.bas

```
start1: ; First task
do
      pulsout red, 5000 ; pulse LED for 50mSec
      pause 1000   ; Each task has a slightly different delay
loop

start2: ; second task
do
      pulsout green, 5000 ; pulse LED for 50mSec
      pause 1100
loop

start3:  ; third task
do
      pulsout yellow, 5000  ; pulse LED for 50mSec
      pause 1200
loop
```

Sample code added to bumper program, to flash Yellow when driving – bumper flash yellow.bas

```
start2:
do
      pulsout yellow, 4000
      pause 500
loop
```

7.2 Extension

TBA

8 Subroutines

8.1 Software

8.1.1 Revise (pre requisites)

- Revise comments, template discipline
- Revise Do/loop, High, Low, Pause
- Revise turns / spin calibration

8.1.2 New introductions

Introduce subroutines - 'Gosub', a separate code routine, terminated by 'Return'.

8.1.3 Tasks

- Subroutines are used to perform repetitive tasks. If we create subroutines to turn 90deg left and right, the time calibration can be adjusted in just one place. These can then be used by a pattern driving program.
- Drive a square
- Drive a figure of eight

Sample code - subroutines to turn left and right – sub pattern.bas

```
;subroutine for left turn
turnl:            ;name of subroutine
      low left, red
      pause 420         ;time for a left turn
      high left,red
return            ;go back to main program

;subroutine for right turn
turnr:            ;name of subroutine
      low right, green
      pause 420         ;time for a right turn
      high right, green
return            ;go back to main program
```

Sample code – drive a figure of eight using the subroutines above – sub pattern.bas

```
do        ;program for figure of eight
      high left,right,red,green
      gosub turnl:            ;turn left
      pause 1000        ;drive for 1 sec
      gosub turnl:            ;turn left
      pause 1000        ;drive for 1 sec
      gosub turnl:            ;turn left
      pause 1000        ;drive for 1 sec
      gosub turnl:            ;turn left
      pause 1000        ;drive for 1 sec
      gosub turnr:            ;turn right
      pause 1000        ;drive for 1 sec
      gosub turnr:            ;turn right
      pause 1000        ;drive for 1 sec
      gosub turnr:            ;turn right
      pause 1000        ;drive for 1 sec
      gosub turnr:            ;turn right
      pause 1000        ;drive for 1 sec
loop
```

9 Binary and memory concepts.

9.1 Software

9.1.1 Revise (pre requisites)

- Revise comments, template discipline , 'save as' file to seperate folder.
- Revise Do/loop, High, Low, Pause

9.1.2 New introductions

- Introduce bytes and word memory, memory locations b0, b1, etc
- Introduce 'inc' increment (add one)
- Introduce binary counting and 0-255 numbers
- Binary count to the port connected to the LEDs (`pinsb`)

9.1.3 Tasks

- Use 'binary chair game' to demonstrate binary values, and multiply/divide by 2 by shifting.
- Binary count on LEDs

Sample code - Binary count on LEDs. Count up to seven? – binary.bas

```
low red,green,yellow ; make LED pins outputs
do
      inc pinsb   ;add one to the LEDs
      pause 1000
loop
```

- Count up and down in binary using bumper switches, watch it on the LEDs

Sample code - Binary count using bumpers – bumper binary.bas

```
Low red,green,yellow ; make LED pins outputs
do
      if lbumper=1 then
            b0=b0+1     ;add one if left bumper
      endif
      if rbumper=1 then
            b0=b0-1     ;subtract one if right bumper
      endif
      pinsb=b0 ;send the result to the LEDs
      pause 200
loop
```

10 Music and tunes.

Not much in this section, but playing ringtones can use a lot of time. Maybe combine with another topic.

10.1 Software

10.1.1 Revise (pre requisites)

- Revise comments, template discipline
- Revise Do/loop
- Sound command with memory control

10.1.2 New introductions

- Tune wizard
- Tune editor

10.1.3 Tasks

Tune wizard

1. Open tune wizard , under the PICAXE menu tab
2. Select output B.7 (spkr) as the 'Piezo pin'
3. Select ' LED flash' output (Yellow is B.1)
4. Use 'Find a tune' link to access tunes website
5. Download 'mixed tunes 1' and unzip to new tunes folder on desktop. (Avoid 'TV tunes' and 'Xmas tunes 'avoid for now, they are a different format)
6. Use 'Open RTTTL.txt file' to select tune, 'generate the BASIC code – yes' and no need to generate the .wav file.
7. Use 'Copy button ' to copy/paste to new program template

Sample code –tune 'abdelaize.txt' should look like this – abdelaize.bas

```
do
tune B.7, 4,%00000010,($F2,$F5,$F9,$02,$44,$45,$47,$45,
$44,$42,$C1,$09,$42,$45,$49,$45,$02,$C9,$07,$40,$44,$47,
$44,$00,$C9,$05,$7B,$42,$45,$42,$3B,$C7,$04,$79,$41,$44,
$40,$39,$C5,$44,$45,$44,$42,$01,$05,$44,$45,$44,$42,$39,
$02,$41,$42,$44,$42,$C2)
loop
```

Tune editor

Using the editor in the tune wizard, try to write a tune of your own in the tune editor!

11 Using the light sensor

11.1 Software

11.1.1 Revise (pre requisites)

- Revise comments, template discipline
- Revise Do/loop
- Sound command with memory control
- Memory and bytes
- If then/ else/ endif

11.1.2 New introductions

- IF/Else/endif using the light sensor as an input
- 'ReadADC' using the 'Vlight' input

11.1.3 Tasks

- Light Tunes. Read the light level using 'Readadc' command, and make a sound dependant on light level . Sounds need a number between 0 to 127 whereas light levels can be 0 to 255, so we divide light level by 2. (b0 and b1 are temporary memory stores).

Sample code – Light Tunes.bas

```
do
      readadc Vlight, b0  ; get light level into byte 0 (b0)
      b1=b0/2           ; make it <127
      sound spkr,(b1,10)      ;make a sound with the result
loop
```

- Read light value using 'readadc' command and show value on LEDs. As the light level can go up to 255, we need to divide the number by 32 to get it back to the 0-7 range for the three LEDs. (b0 and b1 are temporary memory stores).

Sample code - Show light sensor readings on the LEDs – light on LEDs.bas

```
Low red,green,yellow    ;make all LEDs ports as outputs
do
      readadc Vlight, b0      ;fetch light reading
      b1=b0/32                ;divide by 32 so the LEDs can show
      pinsb=b1                ;send it to the LEDs
loop
```

- Follow light. This will require light beam in a dark area. Use the LEDs to show where light and dark areas are.

Sample code – follow edge of light beam – light follow.bas

```
do
      if light=1 then   ;test if light
            high left,red ;turn right
            low right,green
      else
            high right,green ;turn left
            low left,red
      endif
loop
```

11.2 Extension work

- Dairy door chime - Detect changes in light and make a sound when it changes

Sample code - door chime.bas

```
do
      low red,green     ; turn off activity leds
      B2=B1+1     ;make current light+1  the upper limit
      B3=B1-1     ;make current light-1  the lower limit

      readadc vlight,B1 ; read current light level
      if B1>B2 then     ;test if light greater than upper limit
            high red
            sound spkr,(90,10)
      endif

      if B1<B3 then     ;test if light less than lower limit
            high green
            sound spkr,(100,10)
      endif

loop
```

12 Final programming

12.1 Games day. White board patterns.

12.1.1 Software

12.1.2 Revise (pre requisites)

- Revise comments, template discipline
- Revise Do/loop, High, Low, Pause

12.1.3 New introductions

NA

12.1.4 Tasks

Attach whiteboard pen to back end of Rover with bulldog clip or bluetac.
Drive distorted circles on the whiteboard . (Turn more than go straight, to keep it small!)

Sample code - Driving to make orange segments pattern – drive pattern.bas

```
do    ;orange segments
      high left,red          ;turn right….
      pause 2500             ;…for 2.5sec
      low left,red           ;then stop.
      high right,green       ;turn left …
      pause 1500             ;…for 1.5sec
      low right,green        ;then stop
loop
```

12.2 Options for 'take home' programs

1. Bumper programme
2. Light tunes
3. Light follower
4. Door chimes

Three function program to take home **Three prog.bas**

```
; Rover 6
;    Filename:   Three prog
;    Function:  Bumper, light follow, light tune
; ******************************
#no_data
#PICAXE 18m2
;setfreq m32
symbol left = b.3 ; Right forward
symbol right = b.6 ; Left forward
symbol lback = b.4 ; Right reverse
symbol rback = b.5 ; Left reverse
symbol spkr = b.7 ; Speaker
symbol green = b.0 ; LED Green
symbol yellow = b.1 ; LED Yellow
symbol red = b.2  ; LED Red
symbol light = pinc.0 ; Light/dark sensor
symbol Vlight = c.0 ; Variable light sensor
symbol Lbumper = pinc.1 ; Bumper switch left
symbol Rbumper = pinc.2 ; Bumper switch right
symbol nUSB = pinc.5 ; USB not active

high yellow       ;power light
; press a bumper whilst you switch on
; if left bumper pressed then run light tune program
if lbumper = 1 then lighttune

;if right bumper pressed then run light follow program
if rbumper = 1 then lightfollow

;if no bumper pressed then run bumper driving program

bumper:
pause 1000
do
      for w0=1 to 2000
            low rback,lback,red, green
            high right, left, yellow ;drive fwd
            if lbumper=1 then       ;test bumper
                  low left, right   ;stop!
                  high rback, green ; reverse opposite wheel
                  pause 500         ;do this for a while
            endif

            if rbumper=1 then
                  low left, right   ;stop!
                  high lback, red   ; reverse opposite wheel
                  pause 500         ;do this for a while
            endif
      next w0
      low left, right ; intermittent direction change
      high lback
      low yellow ;
      pause 500
loop

; Program to follow light beam
lightfollow:
pause 1000
do
      if light=1 then
            high right,green  ; right on
            low left, red     ; left off
      else
```

```
            low right,green   ;right off
            high left, red    ; left on
      endif
loop

; Program to play sounds from light
lighttune:
low green, yellow, red
do
      readadc vlight,b0 ;get the light level
      pinsb=b0/32 ;send a version to the LEDs
      b0=b0/2     ;divide light by 2 to play the sound
      sound spkr, (b0,10)
loop
```

13 New Concepts (New) and pre-requisites (PR)

Concept	Intro	Build PCB	PCB S/W	Build chassis	Driving	Bumper	Multitask	Subs	Binary	Music	light
Prog editor, Comments, Template, Low,High, MultiNoun, Pause, Do/loop	NEW	PR	PR	PR	PR	PR	PR	PR	PR	PR	PR
Prog Download		NEW	PR	PR	PR	PR	PR	PR	PR	PR	PR
Pulsout		NEW					PR				
If/else/endif			NEW	PR	PR	PR	PR	PR			PR
Play,Sound			NEW							PR	PR
Sleep			NEW								
Reverse				NEW							
For/Next						NEW					
Multitask							NEW				
Subroutine								NEW			
BinaRy, Bytes, Memory									NEW		PR
Increment, Decrement									NEW		
Tune Wiz										NEW	
ReadADC											NEW