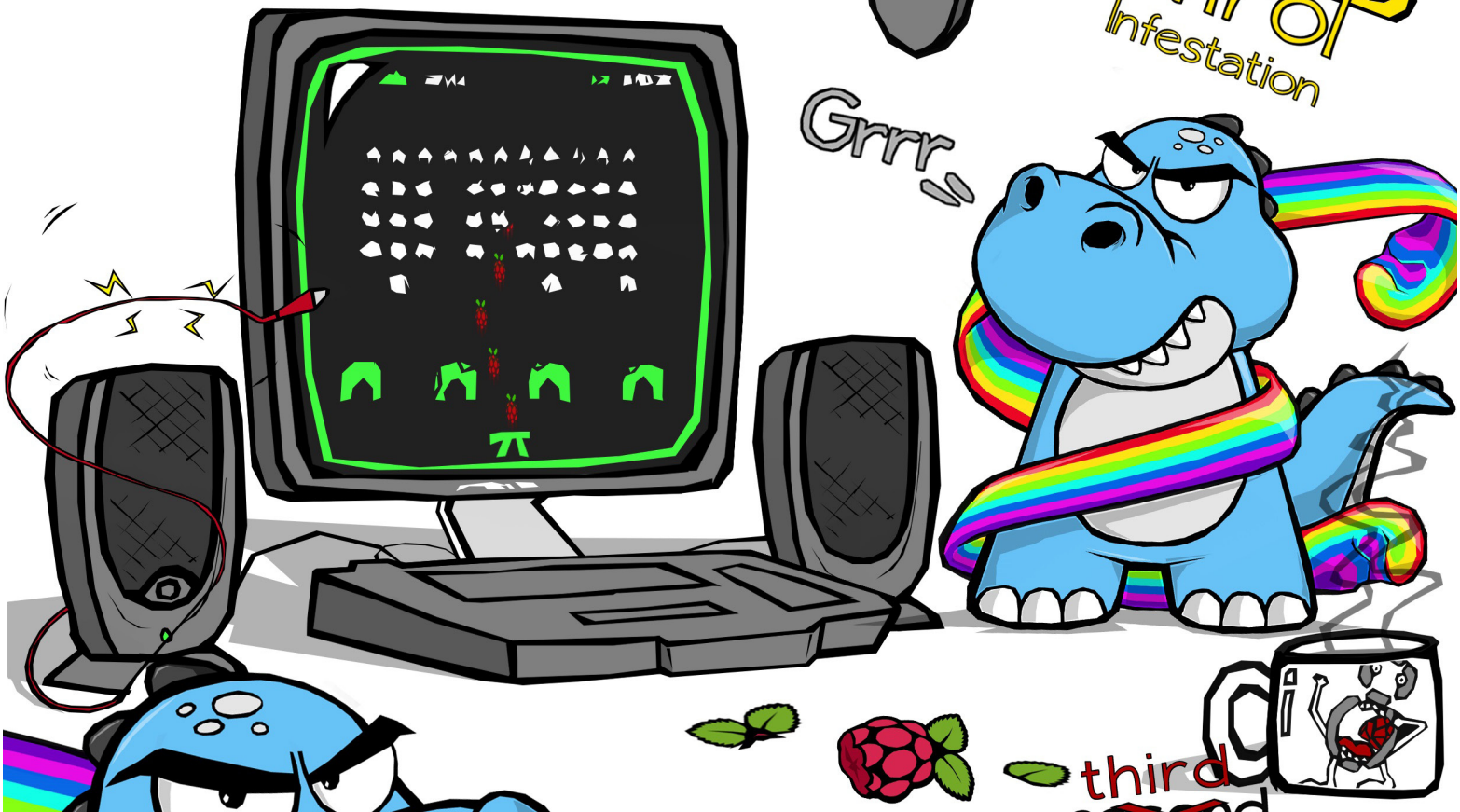
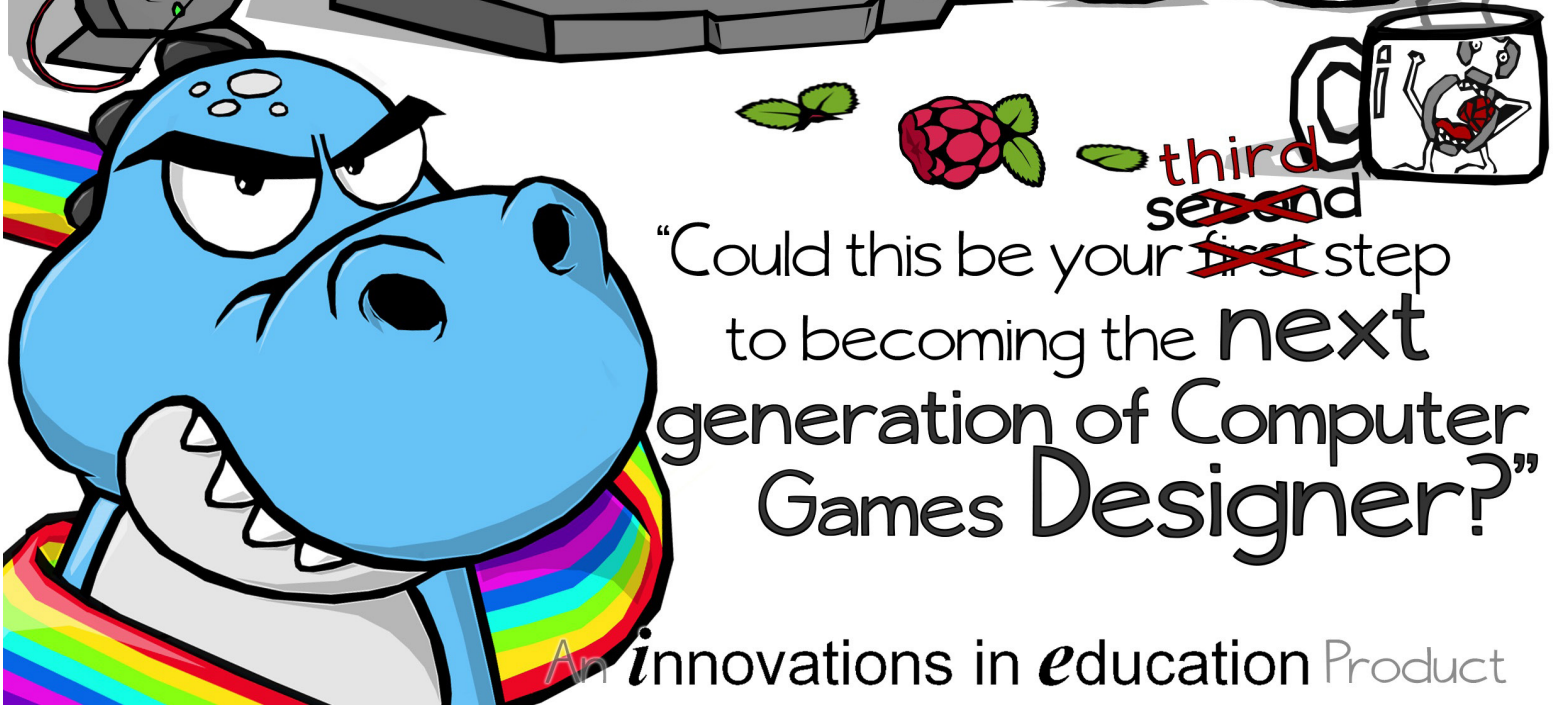
-PiDapter-

GPIO Adapter for The PiTrol





"Could this be your ~~first~~ second third step to becoming the **next generation of Computer Games Designer?**"

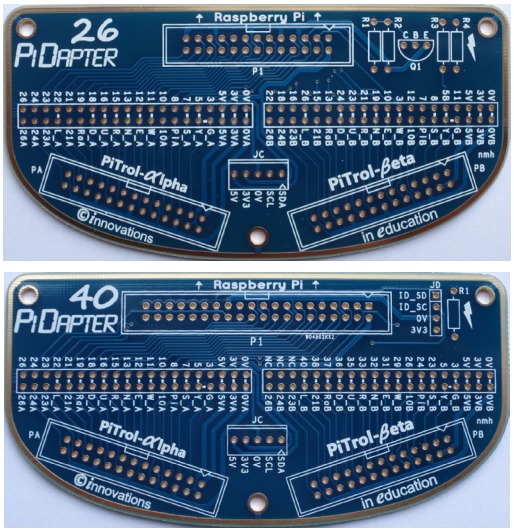
An *innovations in education* Product

Contents

PiDapter - *GPIO Adapter for The PiTrol!*

PiDapter is a **D** shaped GPIO Adapter for your **Raspberry Pi®**!
PiDapter26 is for **Raspberry Pi's** with a ***26*** way GPIO Connector **(Models A & B)**
PiDapter40 is for **Raspberry Pi's** with a ***40*** way GPIO Connector **(Models A+,B+ & 2)**
Connect it by Ribbon Cable to the GPIO Connector on your **Raspberry Pi**
then plug in ***two PiTrol*** games controllers and
write ***two player*** games for the **Raspberry Pi**
PiDapter is unlike other games controller adapters!
PiDapter is designed for ***two player*** games ***written by you!***

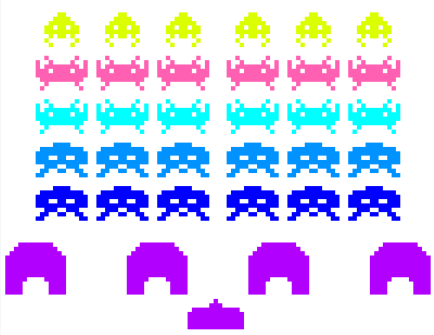
Assemble the *PiDapter* PCB



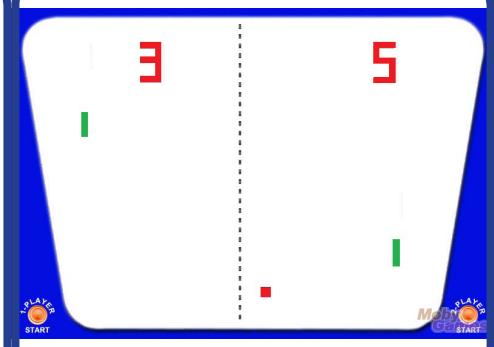
Write the Python game...

After that it is up to you!

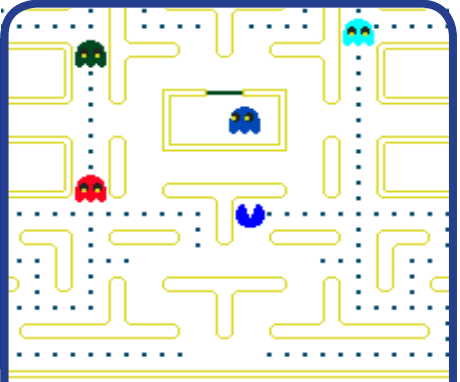
Bring other *two player* Python Games to Life with *PiDapter* !



Convert *two player* Python Games to be Controlled via *PiDapter* !



Enhance *two player* Python Games to use more Controls via *PiDapter* !!



Write new *two player* Python Games especially for *PiDapter* !!!

Soldering and Other Tips

SOLDERING

All components are fitted on the top surface of ***PiDapter's*** Printed Circuit Board (PCB) and soldered on the underside. To assemble the kit you will need the following:

- ***Soldering Iron***
- ***Solder***
- ***Wire Cutters***
- ***Small Pliers***

TIP! - To ease assembly, fit and solder the smallest components first, followed by the next tallest and so on. The Resistors can be fitted either way round but connectors P1, PA, PB and transistor Q1 *MUST BE FITTED AS SHOWN !!!*

TIP! - Solder *just* the 2 opposite corner pins of P1, PA & PB then turn the board over and *double check* they are the right way round *and* are flush with the surface of the PCB before soldering the other pins - *Once more than 2 pins of these components are soldered they are difficult to remove!!!*

PYTHON

***PiDapter* uses the Raspberry Pi 's GPIO so you must be running as "Super User". So when you start the Raspbian IDE GUI type `sudo startx`**

TIP! - Download the *Seven Segments of Pi* Instructions Manual from www.SevenSegmentsOfPi.com and read the Chapter "Running Python Test Program".

Download test program:-

```
PiDapter26_test.py
```

or

```
PiDapter40_test.py
```

from www.PiDapter.com

These test programs are written to run with the Python Editor IDLE. If you try to run it with IDLE3 you will get

```
SyntaxError - invalid syntax
```

To use IDLE3 you would need to modify the program to use Python3 syntax.

TIP! - Read the "Understanding Python", "Python Troubleshooting" and "Python Debugger" Chapters of the *Seven Segments of Pi* Instructions Manual.

SOUND EFFECTS

There can be problems getting Sound Effects to work on the Raspberry Pi as it can depend on what monitor you are using and even what monitor cable you are using!

TIP! - If your Sound Effects don't work, read the "Troubleshooting Sound Effects" Chapter of the *Seven Segments of Pi* Instructions Manual.

FILE HANDLING

Until you make a Backup the only copy of your software is on your SD Card and SD Cards do occasionally fail!

TIP! - Take regular Backups! Read the "Taking Backups and Transferring Files onto your Raspberry Pi" Chapter of the *Seven Segments of Pi* Instructions Manual.

While developing your software you might sometimes wish you could go back to your last working version!

TIP! - Once you have `step1.py` working save the file as `step2.py` before making any more changes. Then you can always go back to `step1.py` if `step2.py` goes all wrong!

PiDapter26 Components

Qty	Part Description	PCB Ref	Assembly Notes, etc
1	PiDapter26 PCB		Fit components from the top, solder on underside
1	Resistor 3k3 ohm	R1	Marked ORANGE, ORANGE, RED, GOLD
1	Resistor 820 ohm	R2	Marked GREY, RED, BROWN, GOLD
1	Resistor 4k7 ohm	R3	Marked YELLOW, PURPLE, RED, GOLD
1	Resistor 680k ohm	R4	Marked BLUE, GREY, YELLOW, GOLD
1	Transistor 2N5551	Q1	! Fit with flat of 'D' shaped plastic to the top
3	Boxed Headers 26 Way	P1,PA,PB	! Fit with cutout towards top of PCB
3	Self Adhesive Feet		Fit on underside of the PCB in positions marked
1	Rainbow Ribbon Cable 26 Way		Connects from PiDapter P1 to Raspberry Pi

NOTE - Connector JC is for I2C and is not supplied as part of the kit

PiDapter40 Components

Qty	Part Description	PCB Ref	Assembly Notes, etc
1	PiDapter40 PCB		Fit components from the top, solder on underside
1	Resistor 680k ohm	R1	Marked BLUE, GREY, YELLOW, GOLD
1	Boxed Header 40 Way	P1	! Fit with cutout towards top of PCB
2	Boxed Headers 26 Way	PA,PB	! Fit with cutout towards top of PCB
3	Self Adhesive Feet		Fit on underside of the PCB in positions marked
1	Rainbow Ribbon Cable 40 Way		Connects from PiDapter P1 to Raspberry Pi

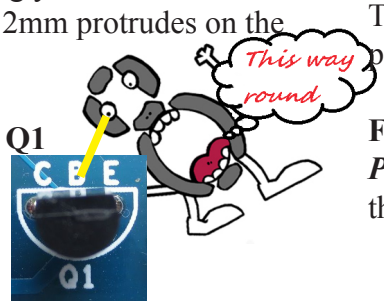
NOTE - Connectors JC & JD are for I2C & ID EEPROM and are not supplied as part of the kit

PiDapter26 Assembly Instructions

Fit and Solder Resistors R1,R2,R3,R4

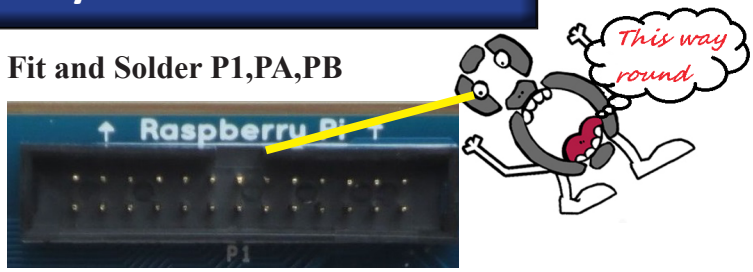
R1 3.3k ohms
ORANGE, ORANGE, RED, GOLD
R2 820 ohms
GREY, RED, BROWN, GOLD
R3 4.7k ohms
YELLOW, PURPLE, RED, GOLD
R4 680k ohms
BLUE, GREY, YELLOW, GOLD
Resistors can be fitted either way round but fitting them all the same way makes it look neater! Once soldered crop the leads using your Wire Cutters so that no more than about 2mm protrudes on the underside.

Fit and Solder Transistor Q1



Q1 is an Transistor with 3 pins, Collector, Base and Emitter which go in holes marked C,B,E. ***IT MUST BE FITTED WITH THE FLAT OF THE 'D' SHAPED PLASTIC TO THE TOP.*** Bend the legs slightly to fit in the holes and push the Transistor down to within 2-3mm of the PCB. Once soldered crop the leads using your Wire Cutters so that no more than about 2mm protrudes on the underside.

Fit and Solder P1,PA,PB



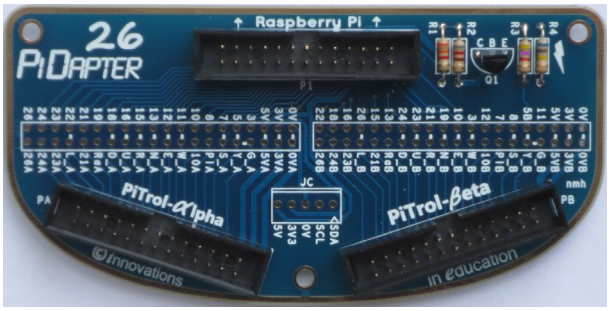
P1,PA and PB are 26 Way Ribbon Cable Connectors which ***MUST BE FITTED WITH THE CUT-OUTS IN THEIR PLASTIC TO THE TOP*** as indicated by the silk screen.

Fit the three Self Adhesive Feet

These fit on the underside of the PCB in the positions marked by three circles on the silk screen.

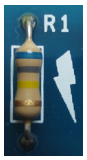
Fully assembled

PiDapter26 PCB when fully assembled looks like this.



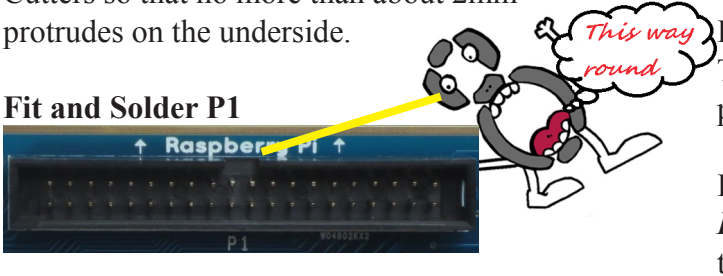
PiDapter40 Assembly Instructions

Fit and Solder Resistor R1



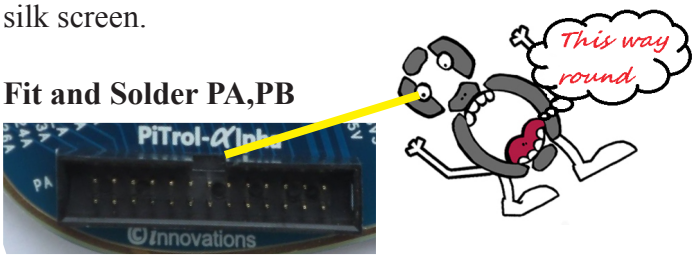
R1 680k ohms
BLUE, GREY, YELLOW, GOLD
Once soldered crop the leads using your Wire Cutters so that no more than about 2mm protrudes on the underside.

Fit and Solder P1



P1 is a 40 Way Ribbon Cable Connector which ***MUST BE FITTED WITH THE CUT-OUTS IN THEIR PLASTIC TO THE TOP*** as indicated by the silk screen.

Fit and Solder PA,PB



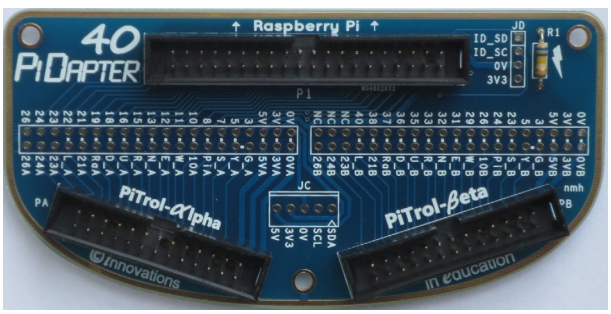
PA and PB are 26 Way Ribbon Cable Connectors which ***MUST BE FITTED WITH THE CUT-OUTS IN THEIR PLASTIC TO THE TOP*** as indicated by the silk screen.

Fit the three Self Adhesive Feet

These fit on the underside of the PCB in the positions marked by three circles on the silk screen.

Fully assembled

PiDapter40 PCB when fully assembled looks like this.



Testing and Troubleshooting

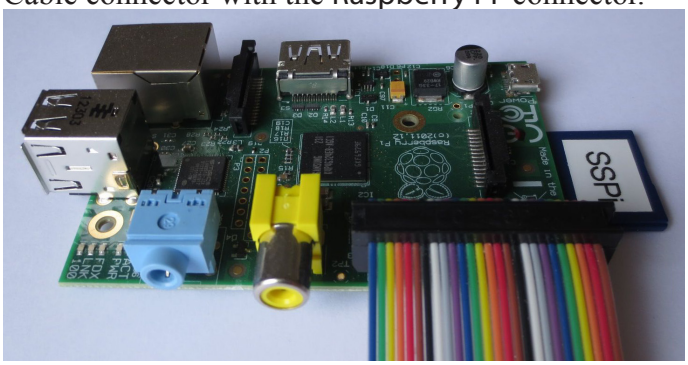
Before attaching ***PiDapter*** to your Raspberry Pi check the following:-

Checks:-
C1) Are **P1**, **PA**,**PB** and **Q1** fitted the correct way round?
C2) Are all pins soldered?
C3) Are there any solder shorts between pins?
C4) Are any Resistor wires touching adjacent components?

If everything looks OK you are ready to attach it to your Raspberry Pi .

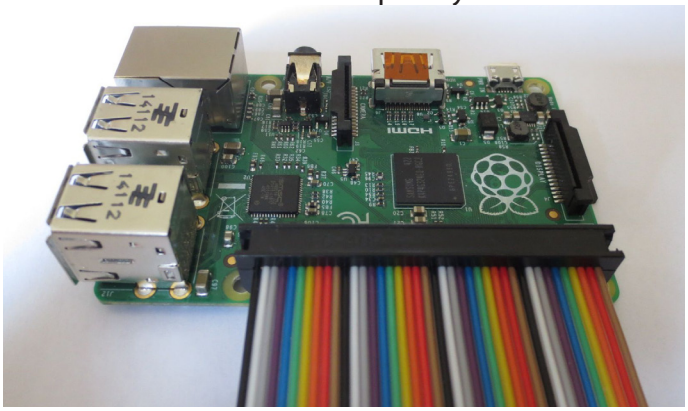
PiDapter26 for Raspberry Pi Models A or B (26 Way GPIO Connector)

Plug one end of the Ribbon Cable into Connector P1 of ***PiDapter26***. Plug the other end into the 26 Way GPIO Header of your Raspberry Pi . The ***PiDapter*** connector is shrouded so the ribbon cable will only fit in it one way round. The Raspberry Pi connector P1 is unshrouded so take care aligning the Ribbon Cable connector with the Raspberry Pi connector.

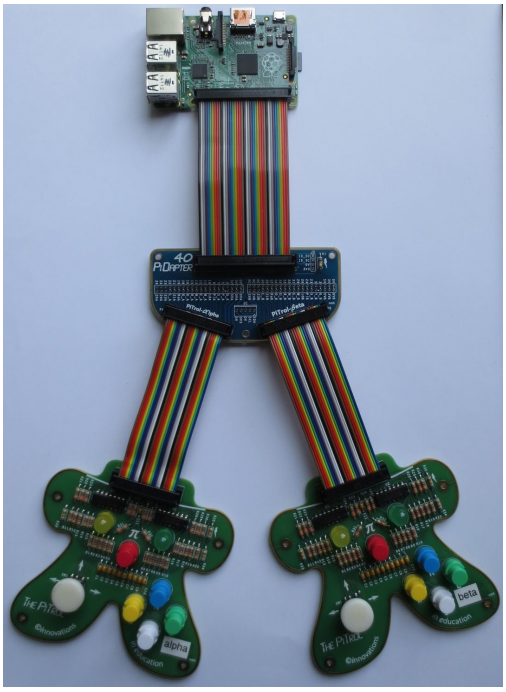


PiDapter40 for Raspberry Pi Models A+,B+or2 (40 Way GPIO Connector)

Plug one end of the Ribbon Cable into Connector P1 of ***PiDapter40***. Plug the other end into the 40 Way GPIO Header of your Raspberry Pi . The ***PiDapter*** connector is shrouded so the ribbon cable will only fit in it one way round. The Raspberry Pi connector P1 is unshrouded so take care aligning the Ribbon Cable connector with the Raspberry Pi connector.



Plug two ***PiTrols*** into the 26 way connectors on ***PiDapter***.



Plug a Raspbian SD Card (or ***Seven Segments of Pi*** "***SSPi***" SD Card) into your Raspberry Pi and power it up.

The PiTrols should power up with both LED's illuminated. If so, you are ready to run the "**Python**" test program as described on the Page 8

If not, follow the troubleshooting tips below to find the assembly problem:-

Troubleshooting Tips:-
T1) Firstly re-check points C1, C2, C3 & C4 listed above.

T2) If your Raspberry Pi fails to boot it could be that your Power Supply cannot supply the extra current of about 20mA needed by two ***PiTrols***. It is recommended that you use a Power Supply that can supply 5V at a current of 1A (1000mA) or more.

T3) If nothing illuminates there could be a problem with the power connection so check the Ribbon Cable is plugged in correctly at both ends and check the soldering of P1 Connector pins 1, 2, 4 & 6.

T4) If the ***PiTrol-Alpha*** or ***PiTrol-Beta*** JoyStick or any of the PushButtons fail to work, check your soldering! Use the ***PiDapter*** Circuit Diagram to help work out which ***PiDapter*** pin are used for which signal.

Python Test Program for PiDapter

Download the Python test program from **www.PiDapter.com**

`PiDapter26_test.py` for ***PiDapter26***
`PiDapter40_test.py` for ***PiDapter40***

Save it in directory:-

`/home/pi/ThePiTrol`

Plug ***PiDapter*** into your Raspberry Pi . Plug in ***PiTrol-Alpha*** and ***PiTrol-Beta*** and run the test program.

If ***PiDapter*** is working correctly the Yellow and Green LED's should alternately flash on ***PiTrol-Alpha*** and ***PiTrol-Beta.*** Then when you press a PushButton Switch or move a JoyStick the "**`Python Name`**" for that button should be printed on the Console Terminal, ***except*** for the "***Raspberry***" and "***Pi***" PushButtons when using ***PiDapter26*** which are ***not*** connected by default.

If any of the switches fail to work read the "***Troubleshooting Tips***" on Page 7 to help find the fault.

Opposite is an excerpt from **`PiDapter26_test.py`**. Highlighted in yellow are the lines you will need to add to your Python Games to control them from ***PiTrol-Beta***!

Below it is an excerpt from **`PiDapter40_test.py`** which is similar but the "***Raspberry***" and "***Pi***" PushButtons can be used.

Driving the LED's
With a single ***PiTrol*** connected
GPIO 3 drives the GreenLED and
GPIO 5 drives the Yellow LED.

With two ***PiTrols*** connected via ***PiDapter26***
GPIO 5 drives ***both*** LED's on ***PiTrol-Alpha*** and
GPIO 5 ***inverted*** drives ***both*** LED's on ***PiTrol-Beta.***
(**GPIO 3** is now used for ***PiTrol-Beta*** "**West**" PushButton so must be configured as an input!)

With two ***PiTrols*** connected via ***PiDapter40***
GPIO 5 drives ***both*** LED's on ***PiTrol-Alpha*** and
GPIO 3 drives ***both*** LED's on ***PiTrol-Beta.***

Read "***Driving the PiTrol LED's***" on Page 22 for more details

Controlling GPIO from Python

In Python the GPIO can be referenced by the Raspberry Pi GPIO Board Header "Physical" Pin Number or by the Broadcom (BCM) Number.

To use the GPIO Board Header Pin Numbers you have to use the Python command

```
GPIO.setmode(GPIO.BOARD)
```

Or to use the BCM Numbers you have to use the Python command

```
GPIO.setmode(GPIO.BCM)
```

Either can be used but the Python test program uses:-

```
GPIO.setmode(GPIO.BOARD)
```

The ***Seven Segments of Pi*** Instructions Manual describes how to write Python Software to configure GPIO as Inputs or Outputs and use them to either sense a PushButton, when an Input, or illuminate an LED when an Output. This Manual can be downloaded from **www.SevenSegmentsOfPi.com**

Just like the ***Seven Segments of Pi***, ***The PiTrol*** has been designed so that when reading an Input in Python, "`True`" means the **PushButton** is pressed and "`False`" means it is *not* pressed. Similarly for Outputs, "`True`" turns the **LED's** 'on' and "`False`" turns it 'off'. (***Note*** that the **LED's** will also be 'on' when the GPIO are unconfigured or configured as inputs hence both **LED's** should be 'on' when the Raspberry Pi is first powered 'on').

```
##############################################################################
# PiDapter26 PiTrol Test Program - PiDapter26_test.py                        #
##############################################################################

# Assign the PiTrol-Beta GPIO the name of the JoyStick direction it controls
B_Up         = 23   #GPIO for Up JoyStick Switch Input
B_Down       = 24   #GPIO for Right JoyStick Switch Input
B_Left       = 26   #GPIO for Left JoyStick Switch Input
B_Right      = 21   #GPIO for Down JoyStick Switch Input
#B_Raspberry = ?    #Raspberry (Centre) JoyStick Switch not connected to GPIO on PiDapter26

# ...and make these GPIO Inputs
GPIO.setup(B_Up,         GPIO.IN)
GPIO.setup(B_Down,       GPIO.IN)
GPIO.setup(B_Left,       GPIO.IN)
GPIO.setup(B_Right,      GPIO.IN)
#GPIO.setup(B_Raspberry, GPIO.IN)

# Assign the PiTrol-Beta GPIO the name of the PushButton Switch it controls
#B_Pi         = ?    #Pi (Red) PushButton Switch not connected to GPIO on PiDapter26
B_North      = 19   #GPIO for Blue North PushButton Switch Input
B_South      =  8   #GPIO for White South PushButton Switch Input
B_East       = 10   #GPIO for Green East PushButton Switch Input
B_West       =  3   #GPIO for Yellow West PushButton Switch Input

# ...and make these GPIO Inputs
#GPIO.setup(B_Pi,     GPIO.IN)
GPIO.setup(B_North, GPIO.IN)
GPIO.setup(B_South, GPIO.IN)
GPIO.setup(B_East,  GPIO.IN)
GPIO.setup(B_West,  GPIO.IN)
```

```
##############################################################################
# PiDapter40 PiTrol Test Program - PiDapter40_test.py                        #
##############################################################################

# Assign the PiTrol-Beta GPIO the name of the JoyStick direction it controls
B_Up         = 35   #GPIO for Up JoyStick Switch Input
B_Down       = 36   #GPIO for Right JoyStick Switch Input
B_Left       = 40   #GPIO for Left JoyStick Switch Input
B_Right      = 33   #GPIO for Down JoyStick Switch Input
B_Raspberry = 37    #Raspberry (Centre) JoyStick Switch

# ...and make these GPIO Inputs
GPIO.setup(B_Up,         GPIO.IN)
GPIO.setup(B_Down,       GPIO.IN)
GPIO.setup(B_Left,       GPIO.IN)
GPIO.setup(B_Right,      GPIO.IN)
GPIO.setup(B_Raspberry, GPIO.IN)

# Assign the PiTrol-Beta GPIO the name of the PushButton Switch it controls
B_Pi         = 24   #Pi (Red) PushButton Switch
B_North      = 32   #GPIO for Blue North PushButton Switch Input
B_South      = 23   #GPIO for White South PushButton Switch Input
B_East       = 31   #GPIO for Green East PushButton Switch Input
B_West       = 29   #GPIO for Yellow West PushButton Switch Input

# ...and make these GPIO Inputs
GPIO.setup(B_Pi,     GPIO.IN)
GPIO.setup(B_North, GPIO.IN)
GPIO.setup(B_South, GPIO.IN)
GPIO.setup(B_East,  GPIO.IN)
GPIO.setup(B_West,  GPIO.IN)
```

GPIO Used by PiDapter26

PiDapter26 allows **two *PiTrols*** to control **two** player Python Games via the Raspberry Pi 's GPIO by "sharing" the ***17*** available GPIO between ***PiTrol-Alpha*** and ***PiTrol-Beta.*** The table below lists the Switches and LED's on both ***PiTrols*** and the GPIO Numbers they connect to ***by default****. The "Python Name" is the name assigned to these GPIO in the Python test program.

PiTrol-Alpha

Component	PCB Ref	Switch	Python Name	GPIO.BOARD	GPIO.BCM	I/O
JoyStick	SW1	↑	A_Up	16	23	Input
JoyStick	SW1	↓	A_Down	18	24	Input
JoyStick	SW1	←	A_Left	22	25	Input
JoyStick	SW1	→	A_Right	15	22	Input
JoyStick	SW1	Centre	A_Raspberry	*	*	Input
PushButton	SW2	Red	A_Pi	*	*	Input
PushButton	SW3	Blue	A_North	13	27	Input
PushButton	SW4	White	A_South	7	4	Input
PushButton	SW5	Green	A_East	12	18	Input
PushButton	SW6	Yellow	A_West	11	17	Input
YellowLED	LED1	LED	LED	5*	3*	Output
GreenLED	LED2	LED	LED	5*	3*	Output

PiTrol-Beta

Component	PCB Ref	Switch	Python Name	GPIO.BOARD	GPIO.BCM	I/O
JoyStick	SW1	↑	B_Up	23	11	Input
JoyStick	SW1	↓	B_Down	24	8	Input
JoyStick	SW1	←	B_Left	26	7	Input
JoyStick	SW1	→	B_Right	21	9	Input
JoyStick	SW1	Centre	B_Raspberry	*	*	Input
PushButton	SW2	Red	B_Pi	*	*	Input
PushButton	SW3	Blue	B_North	19	10	Input
PushButton	SW4	White	B_South	8	14	Input
PushButton	SW5	Green	B_East	10	15	Input
PushButton	SW6	Yellow	B_West	3	2	Input
YellowLED	LED1	LED	LED	/5*	/3*	Output
GreenLED	LED2	LED	LED	/5*	/3*	Output

* With only 17 GPIO available, not all switches and LED's can be used at the same time. ***PiDapter26*** PCB's default wiring means the "***Raspberry***" and "***Pi***" switches are ***un***connected and GPIO 5 controls all LED's, however by changing the default wiring any combination of switches and LED's could be used. For more details see the Chapters on "***Patching the GPIO***" and "***Driving the PiTrol LED's***".

GPIO Used by PiDapter40

PiDapter40 allows **two *PiTrols*** to control **two** player Python Games via the Raspberry Pi 's GPIO by "sharing" the ***26*** available GPIO between ***PiTrol-Alpha*** and ***PiTrol-Beta.*** The table below lists the Switches and LED's on both ***PiTrols*** and the GPIO Numbers they connect to ***by default****. The "Python Name" is the name assigned to these GPIO in the Python test program.

PiTrol-Alpha

Component	PCB Ref	Switch	Python Name	GPIO.BOARD	GPIO.BCM	I/O
JoyStick	SW1	↑	A_Up	16	23	Input
JoyStick	SW1	↓	A_Down	18	24	Input
JoyStick	SW1	←	A_Left	22	25	Input
JoyStick	SW1	→	A_Right	15	22	Input
JoyStick	SW1	Centre	A_Raspberry	19	10	Input
PushButton	SW2	Red	A_Pi	8	4	Input
PushButton	SW3	Blue	A_North	13	27	Input
PushButton	SW4	White	A_South	7	4	Input
PushButton	SW5	Green	A_East	12	18	Input
PushButton	SW6	Yellow	A_West	11	17	Input
YellowLED	LED1	LED	LED	5*	3*	Output
GreenLED	LED2	LED	LED	5*	3*	Output

PiTrol-Beta

Component	PCB Ref	Switch	Python Name	GPIO.BOARD	GPIO.BCM	I/O
JoyStick	SW1	↑	B_Up	35	19	Input
JoyStick	SW1	↓	B_Down	36	20	Input
JoyStick	SW1	←	B_Left	40	21	Input
JoyStick	SW1	→	B_Right	33	13	Input
JoyStick	SW1	Centre	B_Raspberry	37	26	Input
PushButton	SW2	Red	B_Pi	24	8	Input
PushButton	SW3	Blue	B_North	32	12	Input
PushButton	SW4	White	B_South	23	11	Input
PushButton	SW5	Green	B_East	31	6	Input
PushButton	SW6	Yellow	B_West	29	5	Input
YellowLED	LED1	LED	LED	3*	2*	Output
GreenLED	LED2	LED	LED	3*	2*	Output

* With only 26 GPIO available, not all LED's can be controlled individually. ***PiDapter40*** PCB's default wiring connects GPIO 5 to control both ***PiTrol-Alpha*** LED's and GPIO 3 to control both ***PiTrol-Beta*** LED's, however by changing the default wiring LED's could be controlled individually. For more details see the Chapters on "***Patching the GPIO***" and "***Driving the PiTrol LED's***".

Seven Steps to PiDapter-Wormy

The PiTrol Manual gave the Seven Steps to modify **`wormy.py`** into **`PiTrol_wormy_step7.py`**
Here are the Seven Step to modify **`PiTrol_wormy_step7.py`** into **`PiDapter_wormy_step7.py`** to create the two player game ***PiDapter-Wormy***!

In ***PiDapter-Wormy*** , **two** worms compete to "eat" the same apple!
PiTrol-Alpha controls the green worm.
PiTrol-Beta controls the blue worm.
When an apple is "eaten" a new apple appears at a random location.
When a worm goes off the edge of the screen it doesn't die!
Instead, a worm going off to the ***right*** reappears on the ***left***!
A worm going off to the ***left*** reappears on the ***right***!
A worm going off to the ***bottom*** reappears at the ***top***!
And a worm going off to the ***top*** reappears at the ***bottom***!
The winner of the game is the first to "eat" 10 apples!

Before starting, plug ***PiDapter*** into your Raspberry Pi. Plug in ***PiTrol-Alpha*** and ***PiTrol-Beta*** and run **`PiTrol_wormy_step7.py`**. Check that ***PiTrol-Alpha*** controls the game correctly*!

*If you are using ***PiDapter26*** the Red "**Pi**" PushButton is not connected so you will need to make this modification to start the game by pressing the JoyStick "**Up**" instead:-

In the function `def drawPressKeyMsg():`
change the wording of the start message as follows:-

```
pressKeySurf = BASICFONT.render('Press JoyStick "Up" to Play', True, RED)
```

In the function `def checkForKeyPress():`
change the function that checks for a key press to checking if the JoyStick "Up" pressed

```
    if GPIO.event_detected(Up):  # Has JoyStick Up been pressed?
```

SaveAs **`PiDapter_wormy_step1.py`** and you are ready to start! (It's a good idea to give each step a new file name in case you need to go back a steps)

PiDapter_wormy_step1.py

Your first task is to make **`PiDapter_wormy`** ready to be controlled by ***PiTrol-Alpha!***

Change all ***PiTrol*** GPIO names into ***PiTrol-Alpha*** GPIO names
e.g. change '**`Up`**' to '**`A_Up`**', change '**`Down`**' to '**`A_Down`**', etc.
Change the wording of the start message as follows:-

```
pressKeySurf = BASICFONT.render('Press PiTrol-Alpha JoyStick "Up" to Play', True, RED)
```

and adjust the position of the start message as follows:-

```
pressKeyRect.topleft = (WINDOWWIDTH - 380, WINDOWHEIGHT - 30)
```

PiDapter26 only
– delete (or comment out) all references to GPIO **`Raspberry`** and **`Pi`**
– delete all references to GPIO **`GreenLED`** (GPIO 3 is no longer used to drive LED's)
– change all references to GPIO **`YellowLED`** to **`LED`** (GPIO 5 now to drives all LED's)

PiDapter40 only
– change all references to GPIO **`YellowLED`** to **`A_LED`** (GPIO 5 now to drives both ***PiTrol-Alpha*** LED's)
– change all references to GPIO **`GreenLED`** to **`B_LED`** (GPIO 3 now to drives both ***PiTrol-Beta*** LED's)

Run **`PiDapter_wormy_step1.py`** and check the that you can control the game using ***PiTrol-Alpha*** but now the LED's on ***PiTrol-Alpha*** and ***PiTrol-Beta*** should flash alternately.

PiDapter_wormy_step2.py

Your next task is to control the game from ***PiTrol-Beta!***

Copy and paste the yellow highlighted section from **`PiDapter_test.py`** to assign the GPIO pin numbers to the switches

And to detect events on these GPIO, add the following immediately afterwards:-

```
# Detect PiTrol-Beta JoyStick events
GPIO.add_event_detect(B_Up,GPIO.RISING)
GPIO.add_event_detect(B_Down,GPIO.RISING)
```
etc

Then at the start of the main game loop change all these line to use the ***PiTrol-Beta*** GPIO

```
if GPIO.event_detected(B_Left) and A_direction != RIGHT:
if GPIO.event_detected(B_Right) and A_direction != LEFT:
```
etc

Run **`PiDapter_wormy_step2.py`** and check the that you can control the game using ***PiTrol-Beta.*** (although a ***PiTrol-Alpha*** button is still used to start the game!)

PiDapter_wormy_step3.py

Now change the game back to being controlled by ***PiTrol-Alpha*** and change it so the game no longer ends when the worm goes off the edge! Instead make the worm reappear. Make it reappear on the left if it goes off the right, on the right if it goes off the left, on the bottom if it goes off the top and on the top if it goes off the bottom!

At the start of the main game loop change back to use the ***PiTrol-Alpha*** GPIO

```
if GPIO.event_detected(A_Left) and A_direction != RIGHT:
if GPIO.event_detected(A_Right) and A_direction != LEFT:
```
etc

Delete (or comment out) the lines that ends the game if the worm co-ordinate has gone over the edge

```
##          # check if the worm has gone over the edge
##          if wormCoords[HEAD]['x'] == -1 or wormCoords[HEAD]['x'] == CELLWIDTH or
               wormCoords[HEAD]['y'] == -1 or wormCoords[HEAD]['y'] == CELLHEIGHT:
##              sound = DOH                    # ...play the sound
##              sound.play()
##              return # game over
```

And replace with lines to check that if the worm's co-ordinate has gone over the edge it new co-ordinate is on the opposite side

```
# worm going off right appears left, off top appears bottom, etc
if wormCoords[HEAD]['x'] == -1:             # Has it gone off the left?
    wormCoords[HEAD]['x'] = (CELLWIDTH -1) # If so reappear on the right
if wormCoords[HEAD]['x'] == CELLWIDTH:      # Has it gone off the right?
    wormCoords[HEAD]['x'] = 0)             # If so reappear on the left
if wormCoords[HEAD]['y'] == -1: :           # Has it gone off the top?
    wormCoords[HEAD]['y'] = (CELLHEIGHT -1)# If so reappear at the bottom
if wormCoords[HEAD]['y'] == CELLHEIGHT:     # Has it gone off the bottom?
    wormCoords[HEAD]['y'] = 0)             # If so reappear at the top
```

Run **`PiDapter_wormy_step3.py`** and check the that if the worm goes off top, bottom, left or right it reappears on the opposite side!

PiDapter_wormy_step4.py

Your next task is to add a ***second*** worm controlled by ***PiTrol-Beta***!

After `def runGame():` add the starting point and direction for the ***PiTrol-Beta*** worm

```
# PiTrol-Beta worm started a fixed start point half way down right side
startx = CELLWIDTH
starty = (CELLHEIGHT/2)
B_wormCoords = [{'x': startx,     'y': starty},
                {'x': startx - 1, 'y': starty},
                {'x': startx - 2, 'y': starty}]
B_direction = LEFT
```

At the start of the main `while` loop, after the line checking the ***PiTrol-Alpha*** GPIO, add similar lines for ***PiTrol-Beta*** using `B_Left`, `B_Right`, etc and `B_direction, B_nudge`

```
# PiTrol-Beta Joystick GPIO
if GPIO.event_detected(B_Left) and B_direction != RIGHT:
    B_direction = LEFT
if GPIO.event_detected(B_Right) and B_direction != LEFT:
    B_direction = RIGHT
etc

# PiTrol-Beta PushButton GPIO Nudge Up Down Left or Right
if GPIO.event_detected(B_West):
    B_nudge = nLEFT
elif GPIO.event_detected(B_East):
    B_nudge = nRIGHT
etc
```

And make the ***PiTrol-Beta*** worm reappear on the opposite side if it goes over the edge

```
if B_wormCoords[HEAD]['x'] == -1:
    B_wormCoords[HEAD]['x'] = (CELLWIDTH -1)
if B_wormCoords[HEAD]['x'] == CELLWIDTH:
    B_wormCoords[HEAD]['x'] = 0
etc
```

And, after the lines adding new segments to the ***PiTrol-Alpha*** worm, add similar lines for the ***PiTrol-Beta*** worm.

```
# PiTrol-Beta worm moved by adding a segment in the direction it is moving
if B_direction == UP and (B_nudge == NONE or B_nudge == nUP or B_nudge == nDOWN):
    B_newHead = {'x': B_wormCoords[HEAD]['x'], 'y': B_wormCoords [HEAD]['y'] - 1}
elif B_direction == UP and B_nudge == nLEFT: # Nudge Left
    B_newHead = {'x': B_wormCoords[HEAD]['x'] - 1, 'y': B_wormCoords [HEAD]['y']}
elif B_direction == UP and B_nudge == nRIGHT: # Nudge Right
    newHead = {'x': B_wormCoords[HEAD]['x'] + 1, 'y': B_wormCoords [HEAD]['y']}
etc
```

Then this line after the similar one for A

```
B_wormCoords.insert(0, B_newHead)
```

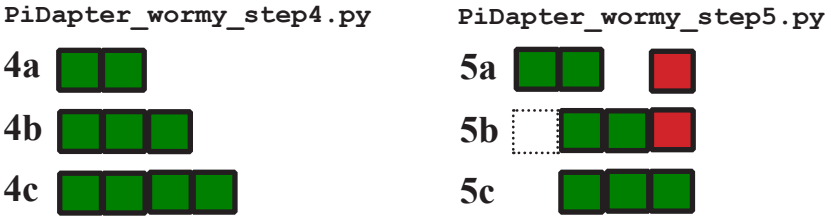
Then this line after the similar one for A

```
drawWorm(B_wormCoords)
```

Run `PiDapter_wormy_step4.py` and check the that both ***PiTrol-Alpha*** and ***PiTrol-Beta*** worms appear...but ***PiTrol-Beta's*** worm will be drawn with an infinite length tail! This will be corrected in step 5!

PiDapter_wormy_step5.py

Your next task is to check if ***PiTrol-Beta*** worm has eaten an apple and remove a tail segment if it has not. This means the tail grows by one segment if it has eaten an apple but will stay the same length if it has not.



In **`step4.py`** as a segment is added to the 'head' (4a to 4b) the 'tail' segment is ***not*** removed.
In **`step5.py`** when as a segment is added to the 'head', the 'tail' segment ***is*** removed if it has ***not*** eaten an apple (5a to 5b) but is ***not*** removed if it ***has*** eaten an apple (5b to 5c)

After the check to see if the ***PiTrol-Alpha*** worm has eaten the apple, add these similar lines to check if the ***PiTrol-Beta*** worm has eaten the apple.

```
# Check if PiTrol-Beta worm has eaten apple
if B_wormCoords[HEAD]['x'] == apple['x'] and B_wormCoords[HEAD]['y'] == apple['y']:
    sound = TWANG  # play the sound
    sound.play()
    FPS = FPS + 1  # speed up each time an apple is eaten
    apple = getRandomLocation() # set a new apple somewhere
else:
    del B_wormCoords[-1] # remove worm's tail segment
```

Run **`PiDapter_wormy_step5.py`** and check the that both ***PiTrol-Alpha*** and ***PiTrol-Beta*** worms appear with ***PiTrol-Beta*** worm's tail the correct length!

PiDapter_wormy_step6.py

Your next task is to make ***PiTrol-Alpha*** worm green and ***PiTrol-Beta*** worm blue.

In function `def drawWorm(wormCoords):` change to

```
pygame.draw.rect(DISPLAYSURF, GREEN, wormInnerSegmentRect)
```

Add a new function `B_drawWorm`, similar to `drawWorm` to draw the blue ***PiTrol-Beta*** worm

```
def B_drawWorm(wormCoords):
    for coord in wormCoords:
        x = coord['x'] * CELLSIZE
        y = coord['y'] * CELLSIZE
        wormSegmentRect = pygame.Rect(x, y, CELLSIZE, CELLSIZE)
        pygame.draw.rect(DISPLAYSURF, DARKBLUE, wormSegmentRect)
        wormInnerSegmentRect = pygame.Rect(x + 4, y + 4, CELLSIZE - 8, CELLSIZE - 8)
        pygame.draw.rect(DISPLAYSURF, BLUE, wormInnerSegmentRect)
```

Add BLUE and DARKBLUE to the list of defined colours

```
BLUE      = (  0,   0, 255)
DARKBLUE  = (  0,   0, 155)
```

Make a change within `def runGame():` to call this new function

```
B_drawWorm(B_wormCoords)
```

Run **`PiDapter_wormy_step6.py`** and check that ***PiTrol-Alpha***'s worm is now green and ***PiTrol-Beta***'s worm is now blue!

PiDapter_wormy_step7.py

Your final task is to change the scoreboard to show both ***PiTrol-Alpha*** and ***PiTrol-Beta*** scores in the same colour as their worm and make the winner the first to 10!

Modify the function displaying ***PiTrol-Alpha***'s score by changing the text and adjusting its position to fit on the screen

```
def drawScore(score):
    scoreSurf = BASICFONT.render('PiTrol-Alpha: %s' % (score), True, GREEN)
    scoreRect = scoreSurf.get_rect()
    scoreRect.topleft = (WINDOWWIDTH - 140, 10)
    DISPLAYSURF.blit(scoreSurf, scoreRect)
```

Add a new function `B_drawScore` to display ***PiTrol-Beta***'s score, similar to the above function but with modified text, adjusted position on the screen and changed colour.

```
def B_drawScore(score):
    scoreSurf = BASICFONT.render('PiTrol-Beta: %s' % (score), True, BLUE)
    scoreRect = scoreSurf.get_rect()
    scoreRect.topleft = (WINDOWWIDTH - 140, 30)
    DISPLAYSURF.blit(scoreSurf, scoreRect)
```

And call this function with ***PiTrol-Beta***'s score (note that the worm starts off 3 squares long and increases in length by 1 square each time an apple is eaten, so the score is equal to the length `B_wormCoords` of minus 3

```
    B_drawScore(len(B_wormCoords) - 3)
```

And modify the main while loop to check if either scores has reached the winning score

```
while ((len(wormCoords) - 3) < winningscore and (len(B_wormCoords) - 3) < winningscore):
```

And set the winning score as a constant before `def main():`

```
winningscore = 10
```

Run **`PiDapter_wormy_step7.py`** and you should have a fully working game of ***PiDapter-Wormy***!!!

PiDapter-Wormy Extras

You may have your own ideas how you could improve ***PiDapter-Wormy*** further, but here are a few suggestions:-

X1) Add a few 'Rotten' Apples which the worms must avoid!

X2) Make it so one worm is not allowed to cross over the tail of the other worm

X3) Add some 'Birds' which roam the screen trying to 'eat' a worm

X4) Eat 'magic' apples to give the worm extra powers, such as enabling the function of extra buttons

X5) Add a worm hole in space. A secret passageway from one part of the screen to another.

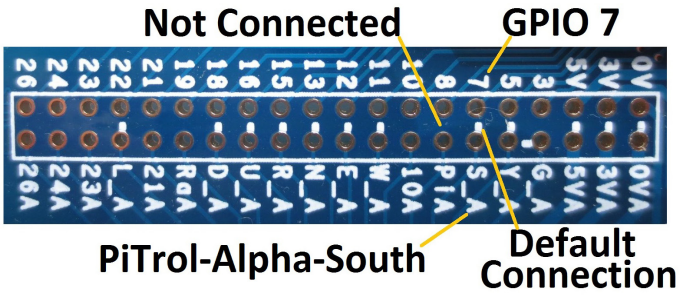
X6) Add a Maze to be negotiated!

X7) Make your own sound effects using free audio editing software 'Audacity'!

Patching the GPIO

Raspberry Pi Models A and B with a 26 Way GPIO Connector have 17 GPIO. Raspberry Pi Models A+, B+ and 2 with a 40 Way GPIO Connector have 26 GPIO. The ***PiTrol*** uses up to 12 GPIO, hence if two or more ***PiTrols*** are connected via ***PiDapter*** there aren't enough GPIO to connect to all Switches and LED's. You therefore need to choose which Switches and LED's to connect! To select which GPIO are connected, ***PiDapter*** has two "patch" connectors, one for ***PiTrol-Alpha*** GPIO and one for ***PiTrol-Beta*** GPIO.

The top row of pins are marked with the GPIO number and the bottom row of pins are marked with the ***PiTrol*** function, so any GPIO could be wired (patched) to any ***PiTrol*** function. However, to make ***PiDapter*** easier to use, it comes pre-wired with default connections which connect the GPIO to the most likely ***PiTrol*** functions as per the Tables on Page 10 & 11.

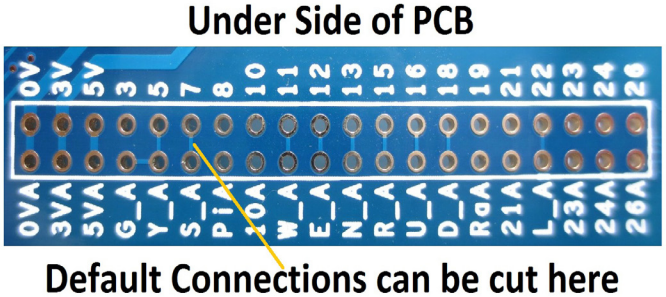


Default Connections

The default connections are ***indicated*** on the PCB by white silk screen lines on the Patch Connector. These lines show for instance that on ***PiDapter26*** GPIO 7 is connected to S_A (S = South PushButton, A = ***PiTrol-Alpha***) but GPIO 8 is ***not*** connected to PiA (Pi = Pi PushButton, A = ***PiTrol-Alpha***)

Disconnecting the Default Wiring

The electrical connections for the default wiring are ***not*** the Silk Screen lines! The ***electrical connections*** for the default wiring are on the ***bottom*** side of the PCB, linking the Patch Connector pin holes. If you want to disconnect any of the default wiring, cut these connections with a scalpel



Reasons to change the Default Wiring

There are 3 main reasons why you might want to change the default wiring.

a) For a Dual ***PiTrol PiDapter26*** game which uses the "***Raspberry***" and/or "***Pi***" PushButtons.

b) When connecting other Raspberry Pi boards such as a Robot Arm that requires different GPIO connected.

c) For a Quad ***PiTrol*** game.

a) The Centre Joystick "***Raspberry***" buttons and the Red "***Pi***" PushButtons are ***not*** connected by default on ***PiDapter26***. If you want to use them in your game you can wire them to a GPIO. For instance, if you want to use the ***Raspberry*** Buttons in your game you could wire GPIO 19 to RaA on the ***PiTrol-Alpha*** Patch Connector and wire GPIO 13 to RaB on the ***PiTrol-Beta*** Patch Connector. But since GPIO 13 and 19 are connected by the default wiring you must also cut the connection between GPIO 13 and N_A on the ***PiTrol-Alpha*** Patch Connector and the connection between GPIO 19 and N_B on the ***PiTrol-Beta*** Patch Connector. In which case the North PushButtons will no longer be connected.

b) If for example you are connecting a ***PiTrol*** and a Robot Arm, the Robot Arm may use some of the GPIO normally used by the ***PiTrol***. In which case disconnect the default connections to ***The PiTrol*** and wire these switches to different GPIO.

c) For a Quad ***PiTrol*** game you must share the GPIO between four ***PiTrol***s! The primary ***PiDapter*** can be a ***PiDapter26*** or a ***PiDapter40*** to match your Raspberry Pi . The two secondary ***PiDapters*** **must** be ***PiDapter26's***.

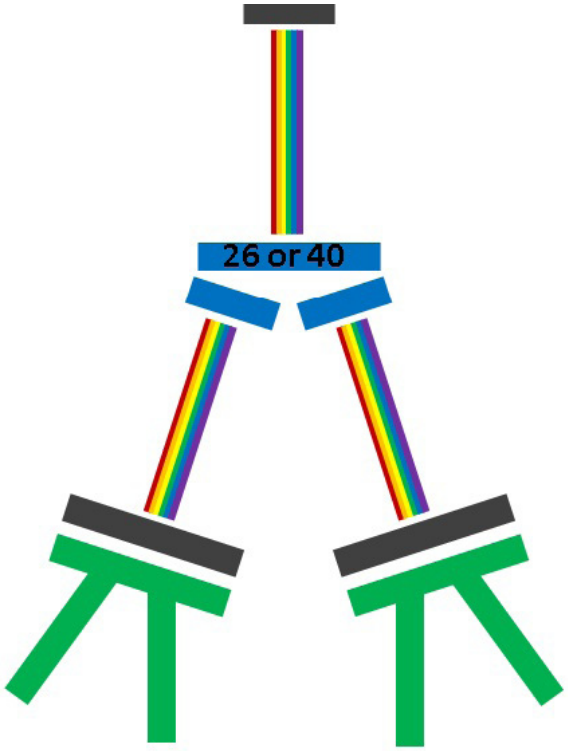
If your primary ***PiDapter*** is a ***PiDapter26*** you will have 4 GPIO available per ***PiTrol.*** You could connect them to the four JoyStick directions (Up,Down,Left,Right) or to the four PushButton Switches (North,South,East,West). The primary ***PiDapter26*** can use the default wiring but on the secondary ***PiDapter26's*** you will need to disconnect the default wiring and add GPIO patch wires.

If your primary ***PiDapter*** is a ***PiDapter40*** you will have 6 GPIO available per ***PiTrol.*** You could connect them to the four JoyStick directions (Up,Down,Left,Right) and to the "***Raspberry***" & "***Pi***" PushButton Switches. The primary ***PiDapter40*** can use the default wiring but on the secondary ***PiDapter26's*** you will need to disconnect the default wiring and add GPIO patch wires.

Applications for PiDapter

Dual PiTrol

PiDapter's main application is to connect **two** ***PiTrols*** to a Raspberry Pi...

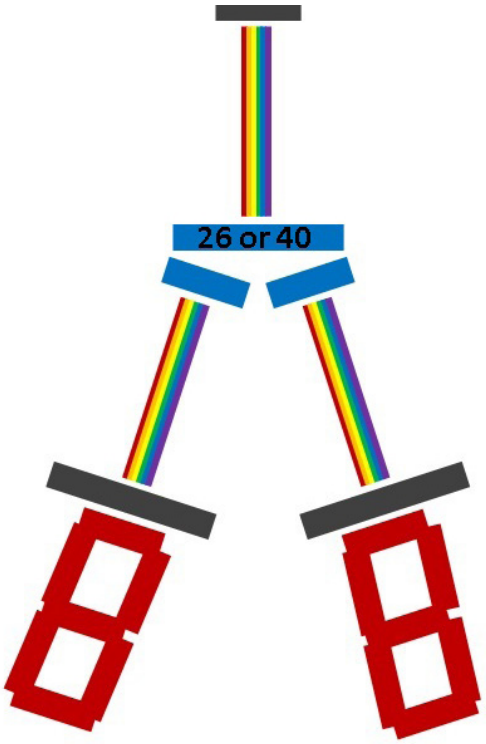


...to control 2 player games...but it can be used for other applications!

Some applications will require GPIO "patching". If so, see the chatper "***Patching the GPIO***" for details of when GPIO patching is necessary and how to patch GPIO on ***PiDapter***.

Dual Seven Segments of Pi

It could be used to connect **two** ***Seven Segments of Pi*** boards to a Raspberry Pi...



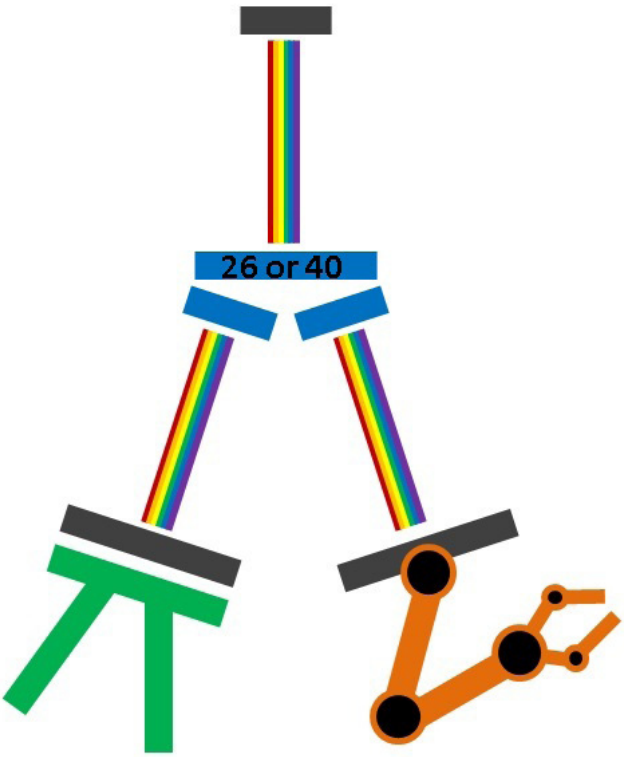
...and write a 2 player version of "***Figure Eight My Pi***", maybe called "***Who Eight all the Pi?***"
...or simply use it as a 2 digit score board!

PiDapter26 and ***PiDapter40*** are designed to allow **two** ***Seven Segments of Pi*** boards to connect like this with no changes to ***PiDapter's*** default connections.

More Applications for PiDapter

PiTrol Controlled Robot Arm

PiDapter can be used to connect any two GPIO boards to the same Raspberry Pi ! For instance you could connect a ***PiTrol*** and a ***Robot Arm***...

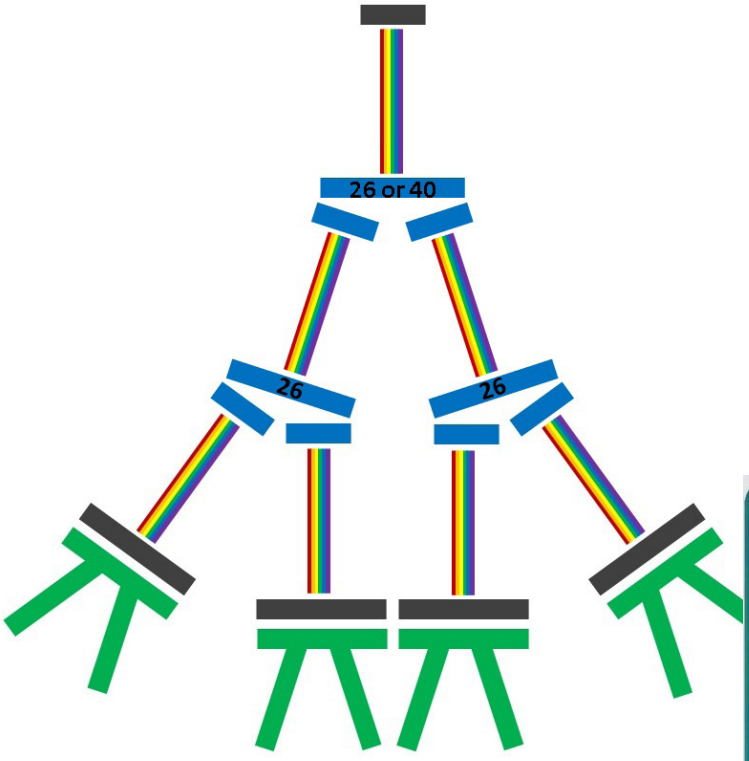


...then write the software so the ***PiTrol*** can be used to control the ***Robot Arm***. Maybe you could even write the software to **record** the ***Robot Arm*** movements so they can be **replayed**!

Depending on the ***Robot Arm*** chosen, it may be necessary to patch the GPIO so the ***Robot Arm*** uses different GPIO from ***The PiTrol.***

Quad PiTrol

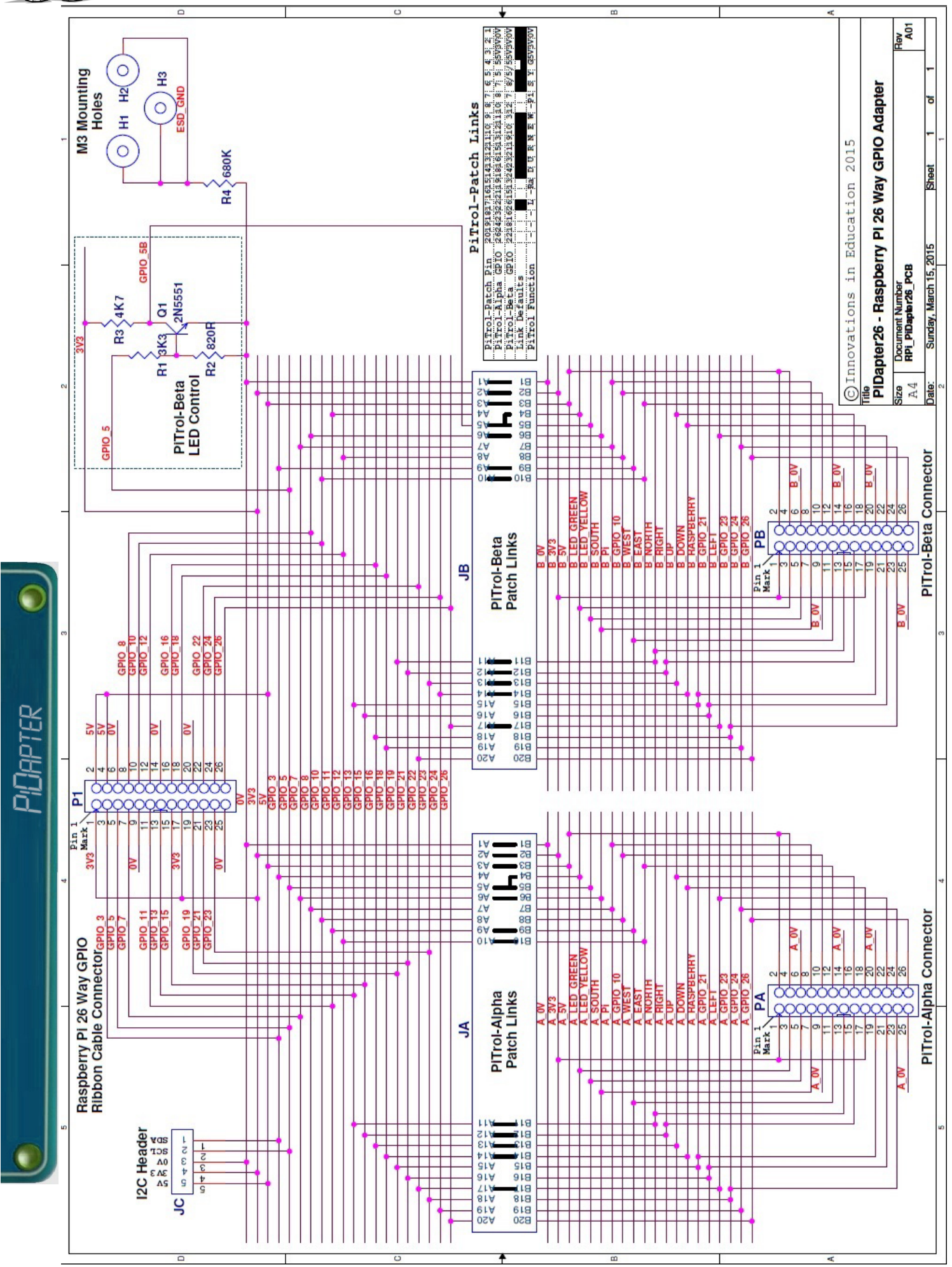
Using **three *PiDapters* (two *PiDapter26*** and **one *PiDapter26*** or ***40)*** you could connect **four *PiTrols*** to a Raspberry Pi...



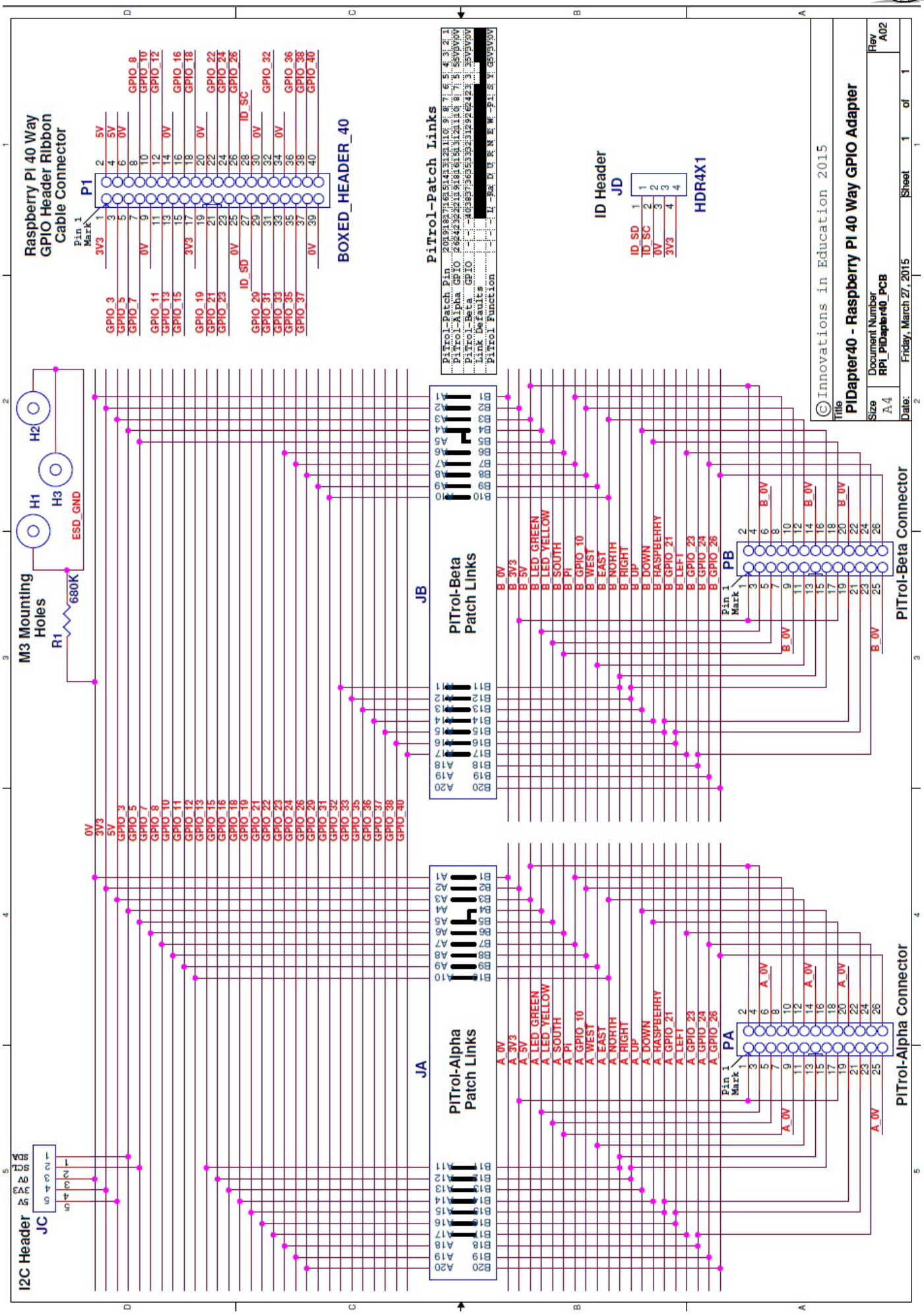
...and write Quad ***PiTrol*** games!

With a limited number of GPIO you will need to choose which ***PiTrol*** controls you require for the game and patch these GPIO on the ***PiDapters.*** For example using ***PiDapter26*** connect the four JoyStick directions on all four ***PiTrols*** or ***PiDapter40*** connect the four JoyStick directions on all four PiTrols and the "***Raspberry***" & "***Pi***" PushButtons.

PiDapter26 Circuit Diagram



PiDapter40 Circuit Diagram



Driving the PiTrol LED's

PiDapter26 connects 8 GPIO to the switches on ***PiTrol-Alpha*** and 8 GPIO to the switches on ***PiTrol-Beta,*** leaving just GPIO 5 to control all four LED's! Although GPIO are binary digital logic signals, they can be set to 3 different states, ***Output-True***, ***Output-False*** and ***Input***. The cicuit on ***PiDapter26*** makes use of this!
GPIO 5 is used to control both LED's on ***PiTrol-Alpha***. Transistor Q1 inverts GPIO 5 to control both LED's on ***PiTrol-Beta***. Hence, when GPIO 5 output is'True' **both** ***PiTrol-Alpha*** LED's will be 'on' and **both** ***PiTrol-Beta*** LED's will be 'off' and vice versa when GPIO 5 output is 'False'. However when GPIO 5 is configured as an **Input** the values of Resistors R1-3 have been chosen so that all four LED's will be 'on'! See the Table below.

	GPIO 5 *input*	GPIO 5 *output True*	GPIO 5 *output False*
PiTrol-Alpha **Yellow LED**	**on**	**on**	**off**
PiTrol-Alpha **Green LED**	**on**	**on**	**off**
PiTrol-Beta **Yellow LED**	**on**	**off**	**on**
PiTrol-Beta **Green LED**	**on**	**off**	**on**

Since the Raspberry Pi powers up with all GPIO configured as ***inputs***, at power up all ***PiTrol*** LED's will be 'on'!

PiDapter40 is more straight forward!
GPIO 5 is used to control both LED's on ***PiTrol-Alpha***
GPIO 3 is used to control both LED's on ***PiTrol-Beta***
The Tables below show how to control the LED's

	GPIO 5 *input*	GPIO 5 *output True*	GPIO 5 *output False*
PiTrol-Alpha **Yellow LED**	**on**	**on**	**off**
PiTrol-Alpha **Green LED**	**on**	**on**	**off**

	GPIO 3 *input*	GPIO 3 *output True*	GPIO 3 *output False*
PiTrol-Beta **Yellow LED**	**on**	**on**	**off**
PiTrol-Beta **Green LED**	**on**	**on**	**off**

Since the Raspberry Pi powers up with all GPIO configured as ***inputs***, at power up all ***PiTrol*** LED's will be 'on'!

Connectors JC (I2C) and JD (ID EEPROM)

Connectors JC and JD are not supplied with the kit but could be fitted for extra functionality.

Connector JC for I2C (*PiDapter26* or *40*)
Connector JC could be fitted to connect an I2C device. I2C uses GPIO 3 & 5 so if using I2C, the GPIO 3 and 5 default wires on the Patch Connectors should be cut. Connector JC can also provides power (5V, 3V3 & 0V) for the I2C device.

I2C can be used to control other electronics connected to the Raspberry Pi, for instance some Robot Arms can be controlled via I2C.

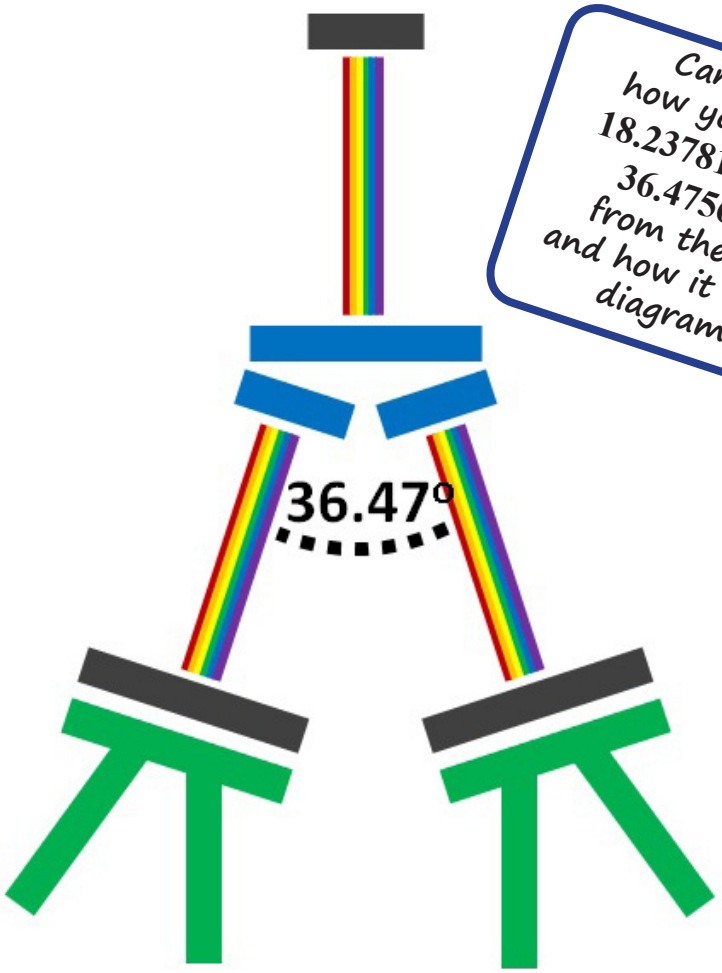
Connector JD for ID EEPROM(*PiDapter40* only)
Connector JD could be fitted to connect an ID EEPROM device. Connector JD can also provide power (3V3 & 0V) for an ID EEPROM device.

ID EEPROMs can be attached to Raspberry Pi Models A+, B+ and 2 only. They can be programmed to indicate to the Raspberry Pi software what electronics is attached.

Mathematical Curiosity

On ***PiDapter***, the Connector for ***PiTrol-Alpha*** is set at an angle of 18.23781306 degrees clockwise from the horizontal. Similarly the Connector for ***PiTrol-Beta*** is set at an angle of 18.23781306 degrees anti-clockwise from the horizontal.

This means that the angle between the Ribbon Cables for ***PiTrol-Alpha*** and ***PiTrol-Beta*** is 36.47562611 degrees! These angles have been chosen as a mathematical curiosity linking ***The PiTrol*** to the number "**π**"!



Can you work out how you get the angles 18.23781306 degrees and 36.47562611 degrees from the number "π" and how it is linked to the diagram opposite?

Instruction Manual Updates

Whilst every effort has been made to ensure these instructions are comprehensive and accurate it is possible there may be errors or omissions. Please let me know of any such errors or omissions via the web site www.PiDapter.com.
Your help in doing so is very much appreciated!

I will publish on the web site any corrections or updates as I become aware of them.

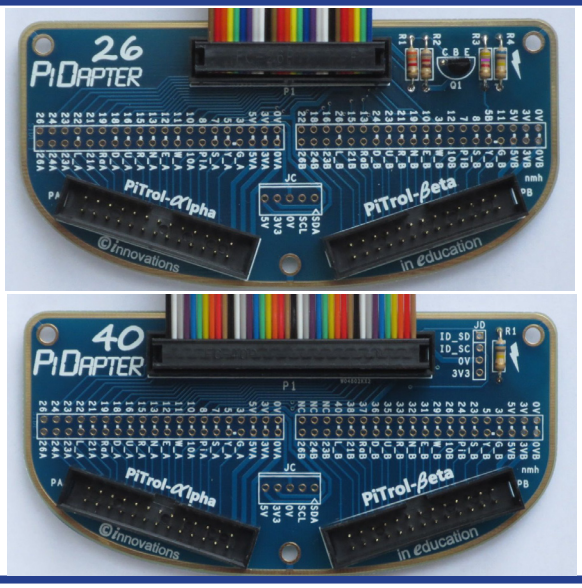
Acknowledgements

I would like to express my gratitude to everyone who has contributed to ***PiDapter***.
In particular I would like to thank:-

Matt Hunt for designing the graphics.
Al Sweigart (http://inventwithpython.com) for the original **`wormy.py`** game released under a "Simplified BSD" license.

- Nevil Hunt innovations in education -

The Complete PiDapter !

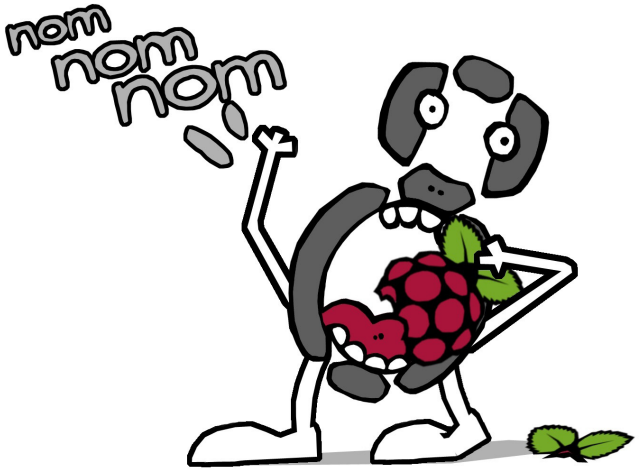


Have Fun while you Learn with PiDapter...

...and this could be your ~~first~~ third step to becoming one of the next generation of Computer Games Designers!

Other Products from innovations in education

Seven Segments of Pi



"Could this be your first step to becoming the next generation of Computer Games Designer?"

www.SevenSegmentsOfPi.com

The PiTrol

"Could this be your ~~first~~ second step to becoming the next generation of Computer Games Designer?"

www.ThePiTrol.com

innovations in education

www.PiDapter.com