**ASSIGNMENT NUMBER: C1**

* **Title**

Controlling the operation of stepper motor using Raspberry Pi /Beagle board circuit.

* **Problem Statement**

Write an application using Raspberry-Pi /Beagle board to control the operation of stepper motor.

* **Objective**

Understanding the connectivity of Raspberry Pi /Beagle board circuit with stepper motor. To understand the actuation.

* **Outcomes**

I will be able to connect Raspberry Pi board circuit with stepper motor.

* **Software & Hardware Requirements**
* Raspberry pi board/ BBB
* Stepper Motor
* Raspbian (OS)
* **Theory**

**Raspberry Pi**

**Introduction**

The Raspberry Pi is a series of credit card-sized single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries. It is a capable little computer which can be used in electronics projects, and for many of the things that your desktop PC does, like spreadsheets, word processing, browsing the internet, and playing games. It also plays high definition video.

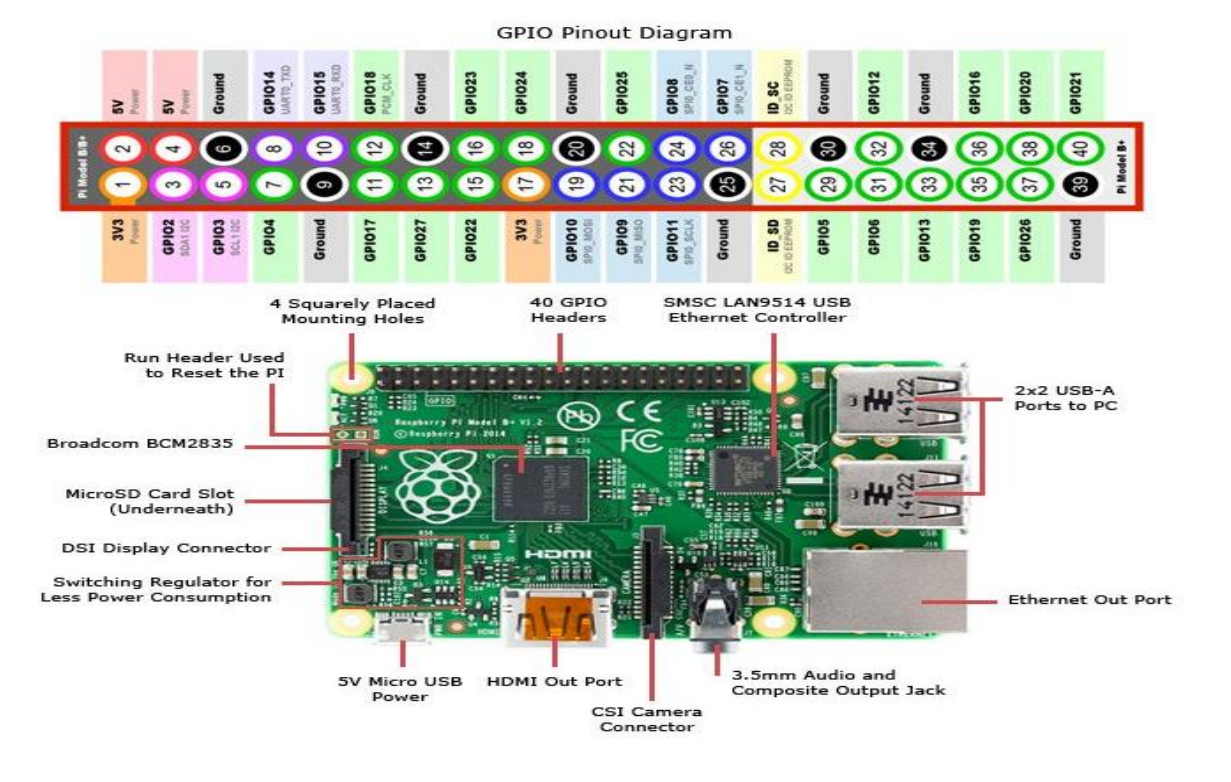
The Raspberry Pi is open hardware, with the exception of the primary chip on the Raspberry Pi, the Broadcom SoC (System on a Chip), which runs many of the main components of the board–CPU, graphics, memory, the USB controller, etc. Many of the projects made with a Raspberry Pi are open and well-documented as well and are things you can build and modify yourself.

The Raspberry Pi was designed for the Linux operating system, and many Linux distributions now have a version optimized for the Raspberry Pi.

One powerful feature of the Raspberry Pi is the row of GPIO (general purpose input/output) pins along the top edge of the board. These pins are a physical interface between the Pi and the outside world. At the simplest level, you can think of them as switches that you can turn on or off (input) or that the Pi can turn on or off (output). Of the 40 pins, 26 are GPIO pins and the others are power or ground pins.

You can program the pins to interact in amazing ways with the real world. Inputs don't have to come from a physical switch; it could be input from a sensor or a signal from another computer or device, for example. The output can also do anything, from turning on an LED to sending a signal or data to another device. If the Raspberry Pi is on a network, you can control devices that are attached to it from anywhere and those devices can send data back. Connectivity and control of physical devices over the internet is a powerful and exciting thing, and the Raspberry Pi is ideal for this.

Raspberry Pi Board with GPIO:



Technical Specification

- Broadcom BC M28 37 64bit AR Mv7 Quad Core Processor powered Single Board

- Computer running at 1.2GHz

- 1GB RAM

- BCM43143 WiFi on board

- Bluetooth Low Energy (BLE) on board

- 40pin extended GPIO

- 4 x USB 2 ports

- 4 pole Stereo output and Composite video port

- Full size HDMI

- CSI camera port for connecting the Raspberry Pi camera

- DSI display port for connecting the Raspberry Pi touch screen display

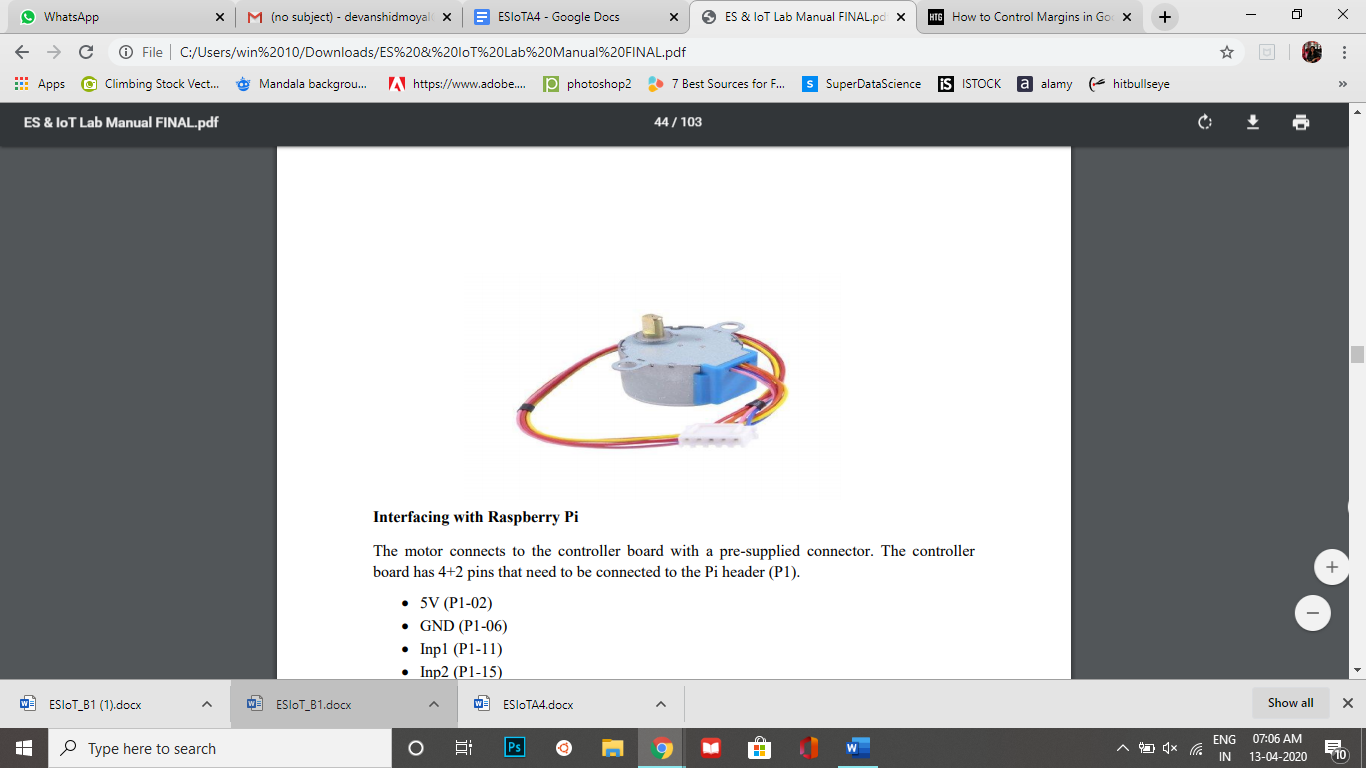
- Micro SD port for loading your operating system and storing data

- Upgraded switched Micro USB power source (now supports up to 2.4 Amps)

- Expected to have the same form factor has the Pi 2 Model B, however the LEDs will Change position.

**Stepper Motor**

A stepper motor is an electromechanical device which converts electrical pulses into discrete mechanical movements. The shaft or spindle of a stepper motor rotates in discrete step increments when electrical command pulses are applied to it in the proper sequence. The motors rotation has several direct relationships to these applied input pulses. The sequence of the applied pulses is directly related to the direction of motor shafts rotation. The speed of the motor shafts rotation is directly related to the frequency of the in put pulses and the length of rotation is directly related to the number of input pulses applied.One of the most significant advantages of a stepper motor is its ability to be accurately controlled in an open loop system. Open loop control means no feedback information about position is needed. This type of control eliminates the need for expensive sensing and feedback devices such as optical encoders. Your position is known simply by keeping track of the input step pulses.

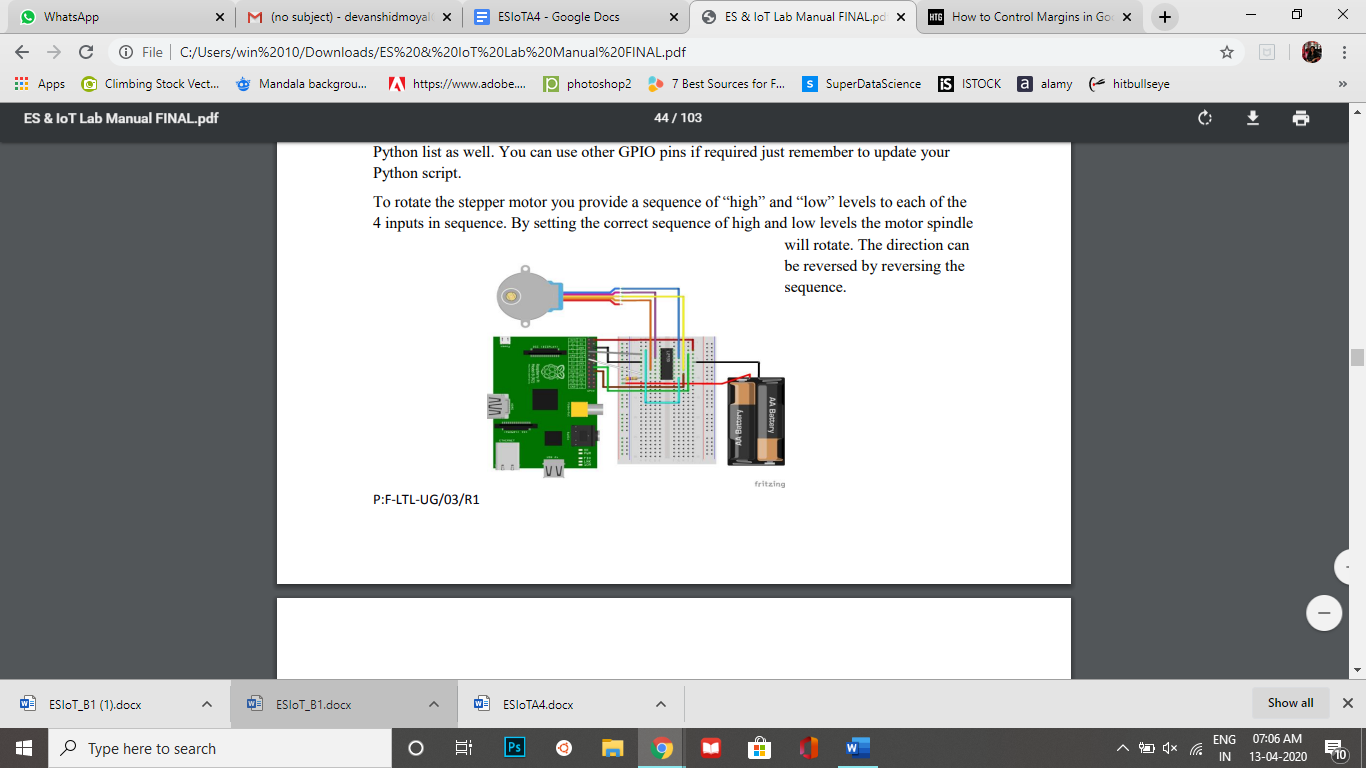


**Interfacing with Raspberry Pi**

Interfacing with Raspberry Pi The motor connects to the controller board with a pre-supplied connector. The controller board has 4+2 pins that need to be connected to the Pi header (P1).

* 5V (P1-02)
* GND (P1-06)
* Inp1 (P1-11)
* Inp2 (P1-15)
* Inp3 (P1-16)
* Inp4 (P1-18)

The P1-XX references above represent the Pi header pins used. These are defined in the Python example below in the StepPins list so if you use different pins be sure to update the Python list as well. You can use other GPIO pins if required just remember to update your Python script. To rotate the stepper motor you provide a sequence of “high” and “low” levels to each of the 4 inputs in sequence. By setting the correct sequence of high and low levels the motor spindle will rotate. The direction can be reversed by reversing the sequence.



**Code:**

#!/usr/bin/python

# Import required libraries

import sys

import time

import RPi.GPIO as GPIO

# Use BCM

GPIO references

# instead of physical pin numbers

GPIO.setmode(GPIO.BCM)

# Define GPIO signals to use

# Physical pins 11,15,16,18

# GPIO17,GPIO22,GPIO23,GPIO24

StepPins = [17,22,23,24]

# Set all pins as output

for pin in StepPins:

print "Setup pins"

GPIO.setup(pin,GPIO.OUT)

GPIO.output(pin, False)

# Define advanced sequence

# as shown in manufacturers datasheet

Seq = [[1,0,0,1],

[1,0,0,0],

[1,1,0,0],

[0,1,0,0],

[0,1,1,0],

[0,0,1,0],

[0,0,1,1],

[0,0,0,1]]

StepCount = len(Seq)

StepDir = 1 # Set to 1 or 2 for clockwise

# Set to -1 or -2 for anti-clockwise

# Read wait time from command line

if len(sys.argv)>1:

WaitTime = int(sys.argv[1])/float(1000)

else:

WaitTime = 10/float(1000)

# Initialise variables

StepCounter = 0

# Start main loop

while True:

print StepCounter,

print Seq[StepCounter]

for pin in range(0,4):

xpin=StepPins[pin]# Get GPIO

if Seq[StepCounter][pin]!=0:

print " Enable GPIO %i" %(xpin)

GPIO.output(xpin, True)

else:

GPIO.output(xpin, False)

StepCounter += StepDir

# If we reach the end of the sequence

# start again

if (StepCounter>=StepCount):

StepCounter = 0

if (StepCounter<0):

StepCounter = StepCount+StepDir

# Wait before moving on

time.sleep(WaitTime)

* **Conclusion**

Thus we have successfully established an interface between Raspberry Pi and Stepper Motor.