Interfacing ADC with R-PI

e-Yantra Team

July 8, 2015

Contents

1 Objective

The objective is to Interface the ADC MCP3008 to Raspberry Pi with a Sharp IR Sensor as Raspberry Pi has no built in analogue inputs.Then the RPi is interfaced with 2 MC3008 ADC's where Sharp IR sensor and White line sensor are connected.

2 Prerequisites

One should have:

- To know how to access the pins of an R-Pi.
- Some basic information regarding SPI protocol.
- Python programming skills.

3 Hardware Requirement

1. Raspberry Pi (I will be using Version 2 Model B)
2. Power adapter(5V)
3. MCP3008
4. Sharp IR Sensor
5. White Line Sensor
6. Connecting wires
7. Bread board

4 Software Requirement

1. MobaXterm (for windows user)
2. PyScripter (version 2.7 or above)

5 Theory and Description

5.1 MCP 3008

- The MCP3008 is a 10bit 8-channel Analogue-to-digital converter (ADC).[1]
- It is cheap, easy to connect and doesnt require any additional components.

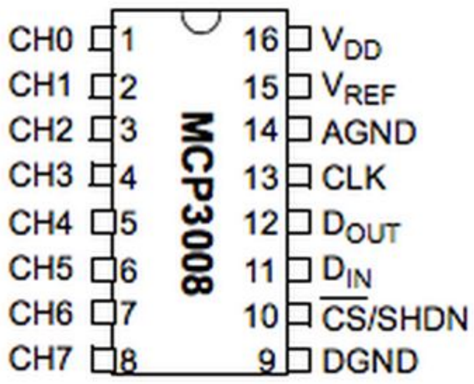

Figure 1: MCP3008

- It uses the SPI bus protocol which is supported by the Pis GPIO header.
- The experiment explains how to use an MCP3008 [5] device to provide 8 analogue inputs which you can use with a range of sensors.
- In the circuit I use my MCP3008 to read a Sharp IR sensor.

5.2 SPI

- The Serial Peripheral Interface (SPI) is a communication protocol used to transfer data between micro-computers like the Raspberry Pi and peripheral devices.[2]
- These peripheral devices may be either sensors or actuators.
- In this example, we will be learning to use an Analog to Digital Converter (ADC) sensor.
- An analog to digital sensor takes an analog voltage and converts it into a digital number that can be understood by the Raspberry Pi.

SPI uses 4 separate connections to communicate with the target device. These connections are:

1. Serial clock (CLK)
2. Master Input Slave Output (MISO)
3. Master Output Slave Input (MOSI) and
4. Chip Select (CS).

- The **Clock pin** sense pulses at a regular frequency, the speed at which the Raspberry Pi and SPI device agree to transfer data to each other. For the ADC, clock pulses are sampled on their rising edge, on the transition from low to high.

- The **MISO** pin is a data pin used for the master (in this case the Raspberry Pi) to receive data from the ADC. Data is read from the bus after every clock pulse.[3]

- The **MOSI** pin sends data from the Raspberry Pi to the ADC. The ADC will take the value of the bus on the rising edge of the clock. This means the value must be set before the clock is pulsed.

- Finally, the **Chip Select line** chooses which particular SPI device is in use. If there are multiple SPI devices, they can all share the same CLK, MOSI, and MISO. However, only the selected device has the Chip Select line set low, while all other devices have their CS lines set high. A high Chip Select line tells the SPI device to ignore all of the commands and traffic on the rest of the bus.

6 Experiment

6.1 Interfacing an ADC with RPi using a Sharp IR Sensor

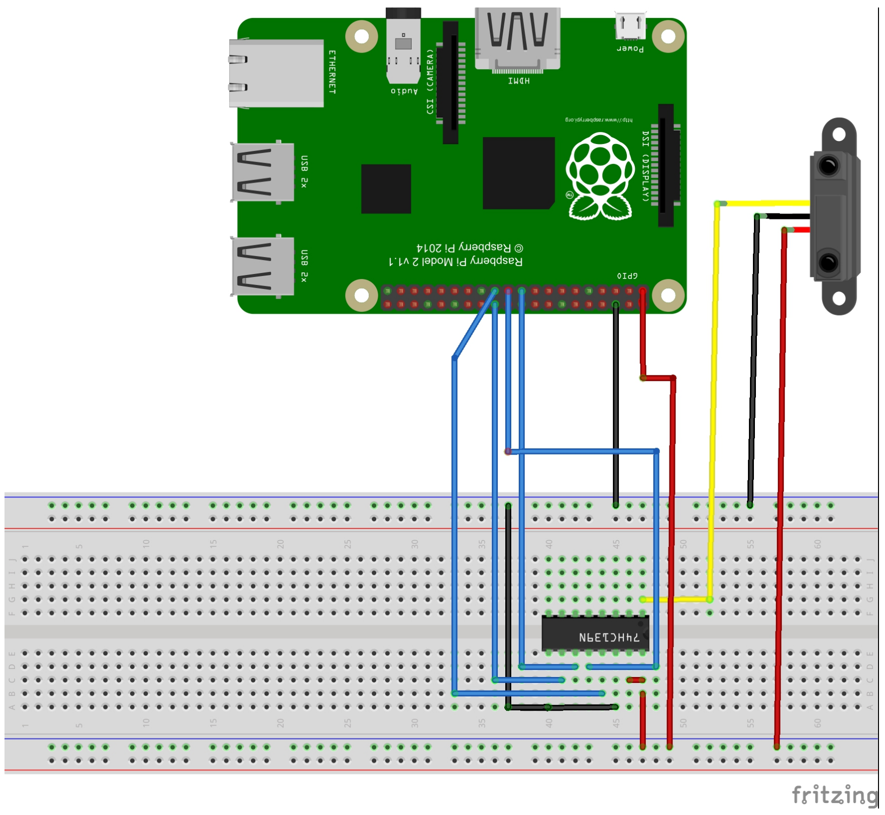

Figure 2: Circuit on Bread Board

- The pin connections are shown in the Circuit above.
- The Connections are showed in the Circuit Schematic in the next figure.
- Here I have used a single ADC MCP3008 interfaced with RPi to which a Sharp IR Sensor is connected.
- In the next Experiment I will use 2 ADC's which will be interfaced with RPi using the 2 chip select pins CE0 and CE1.

6.2 Circuit Schematic

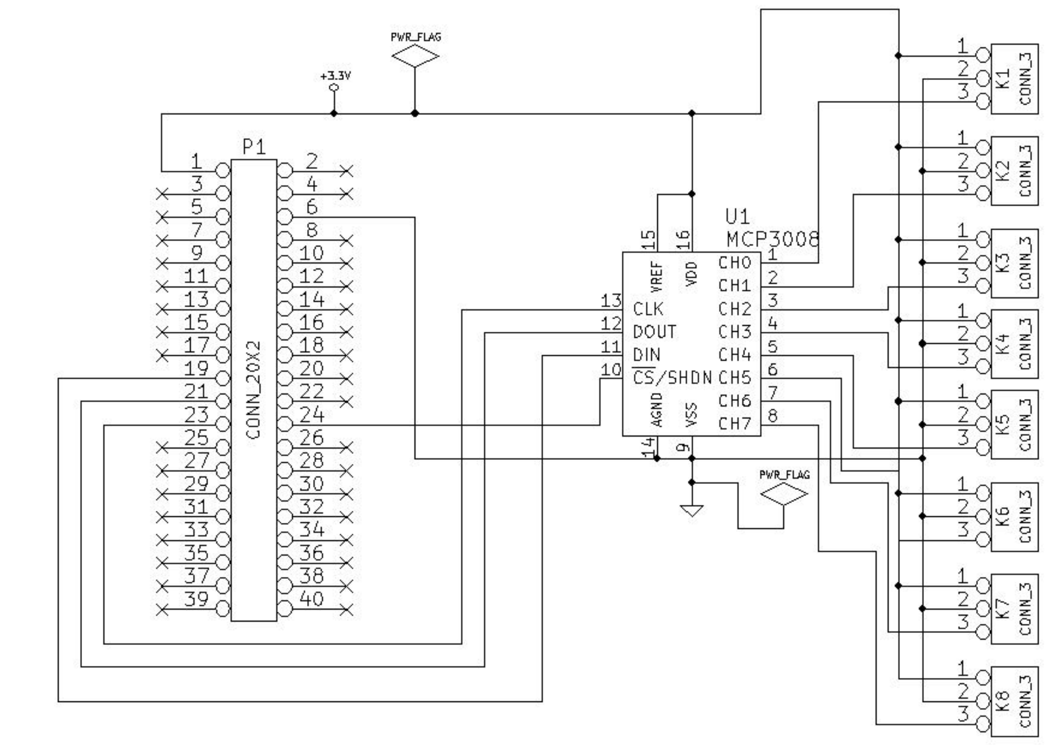

Figure 3: Circuit Schematic

Code

```
import os
import spidev # module to control spi devices
import time

# Open SPI bus
spi = spidev.SpiDev()# to create spi object
spi.open(0,0)#Clock polarity,Clock Phase

# Function to read SPI data from MCP3008 chip
# Channel must be an integer 0-7
# Function name :ReadChannel
# Input : channel
# Output : data
# Example call: ReadChannel(channel)
def ReadChannel(channel):
        #Performs SPI Transaction and CS will be held active
        adc = spi.xfer2([1,(8+channel)<<4,0])
        data = ((adc[1]&3) << 8)
        return data

# Function to convert data to voltage level,
# rounded to specified number of decimal places.
# Function name :ConvertVolts
# Input : data,places
# Output : volts
# Example call: ConvertVolts(data,places)
def ConvertVolts(data,places):
        volts = (data * 3.3) / float(1023)
        volts = round(volts,places)
        return volts

# This Function calculates the actual distance in millimeters(mm)
# from the input
# Function name :Sharp_GP2D12_estimation
# Input : adc_reading
# Output : distanceInt
# Example call: Sharp_GP2D12_estimation(adc_reading)
def Sharp_GP2D12_estimation(adc_reading):
       distance =(10.00*(2799.6*(1.00/(pow(adc_reading,1.1546)))))
       distanceInt = distance
       if distanceInt >800:
       distanceInt=800
```

```
        return distanceInt

# Define sensor channels
sharp_channel =0
white_channel =1

# Define delay between readings
delay =0.5

try:
while True:
        # Read the sharp sensor data
        sharp_level = ReadChannel(sharp_channel)
        sharp_volts = ConvertVolts(sharp_level,3)
        value = Sharp_GP2D12_estimation(sharp_level)

        # Read the white line sensor data
        white_level=ReadChannel(white_channel)
        white_volts = ConvertVolts(white_level,3)

        # Print out results
        print "--------------------------------------------"
        print("sharp: D{} (A{}V)".format(sharp_level,sharp_volts))
        print("sharp: (Distance {} cm)".format(value))

        print "--------------------------------------------"
        print("white: D{} (A{}V)".format(white_level,white_volts))

        # Wait before repeating loop
        time.sleep(delay)
        except KeyboardInterrupt:
pass
```

6.3 PCB Designing

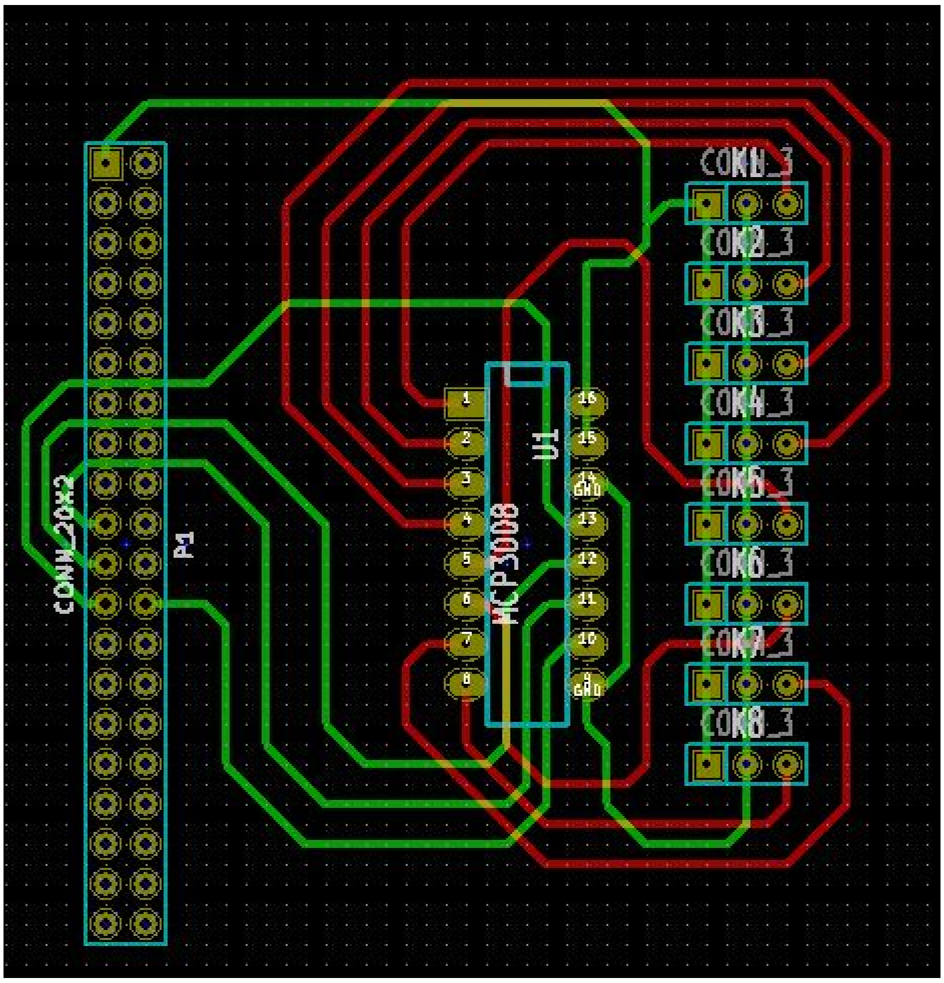

Figure 4: PCB

7 Exercise

7.1 Interfacing 2 ADC's with RPi using a Sharp IR and White line sensor

- The pin connections are shown in the Circuit Schematic.
- Here I have used 2 MCP3008 ADC's interfaced with RPi.
- In the this experiment I have used Sharp IR Sensor and White line sensor.
- 2 ADC's can be interfaced with RPi mainly because of the 2 chip select pins CE0 and CE1 in RPi.

7.2 Circuit Schematic

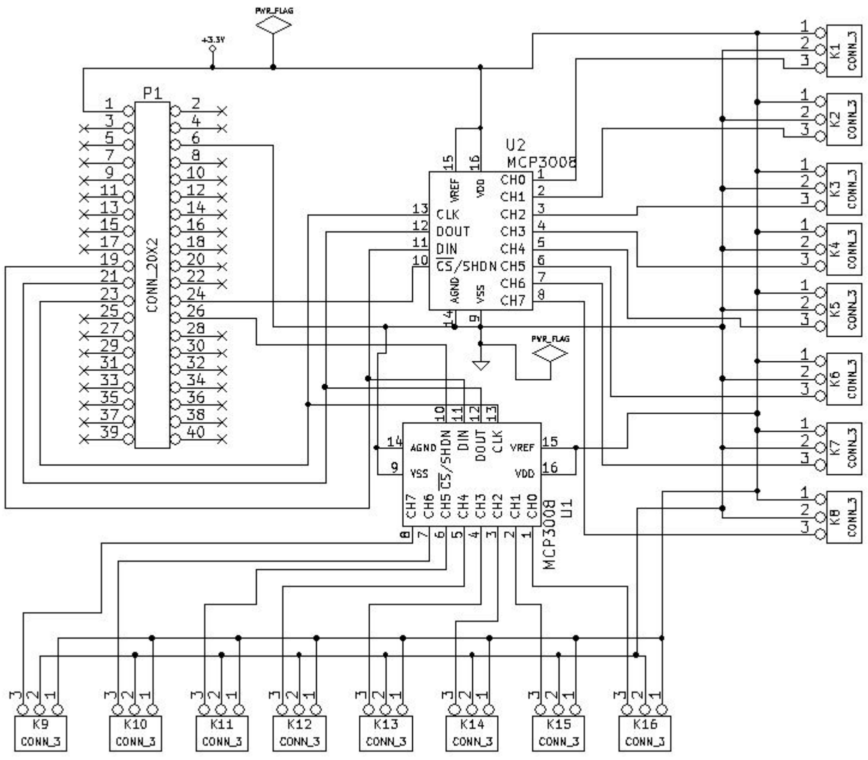

Figure 5: Circuit Schematic

Code

```
import os
import spidev # module to control spi devices
import time
import RPi .GPIO as GPIO # module to control Pi GPIO channels

# Open SPI bus
spi = spidev.SpiDev()# to create spi object
spi.open(0,0)#Clock polarity,Clock Phase

# Function to read SPI data from MCP3008 chip
# Channel must be an integer 0-7
# Function name :ReadChannel
# Input : channel
# Output : data
# Example call: ReadChannel(channel)
def ReadChannel(channel):
        #Performs SPI Transaction and CS will be held active
        adc = spi.xfer2([1,(8+channel)<<4,0])
        data = ((adc[1]&3) << 8)
        return data

# Function to convert data to voltage level,
# rounded to specified number of decimal places.
# Function name :ConvertVolts
# Input : data,places
# Output : volts
# Example call: ConvertVolts(data,places)
def ConvertVolts(data,places):
        volts = (data * 3.3) / float(1023)
        volts = round(volts,places)
        return volts

# This Function calculates the actual distance in millimeters(mm)
# from the input
# Function name :Sharp_GP2D12_estimation
# Input : adc_reading
# Output : distanceInt
# Example call: Sharp_GP2D12_estimation(adc_reading)
def Sharp_GP2D12_estimation(adc_reading):
       distance =(10.00*(2799.6*(1.00/(pow(adc_reading,1.1546)))))
       distanceInt = distance
       if distanceInt >800:
        distanceInt=800
        return distanceInt

# Define sensor channels
sharp_channel =0
white_channel =1

# Define delay between readings
delay =0.5

try:
while True:
       GPIO.setmode(GPIO.BOARD)# set up GPIO output channel
       GPIO.setup(24, GPIO.OUT)# set up CE0 Pin
       GPIO.setup(26, GPIO.OUT)# set up CE1 Pin

       GPIO.output(24,GPIO.LOW)# Enable CE0 Pin
       GPIO.output(26,GPIO.HIGH)# Disable CE1 Pin


       # Read the sharp sensor data
       sharp_level = ReadChannel(sharp_channel)
       sharp_volts = ConvertVolts(sharp_level,3)
       value = Sharp_GP2D12_estimation(sharp_level)

       GPIO.output(26,GPIO.LOW)# Enable CE1 Pin
       GPIO.output(24,GPIO.HIGH)# Disable CE0 Pin

       # Read the white line sensor data
       white_level=ReadChannel(white_channel)
       white_volts = ConvertVolts(white_level,3)

       # Print out results
       print "————————————————————————————————————————"
       print("sharp: D{} (A{}V)".format(sharp_level,sharp_volts))
       print("sharp: (Distance {} cm)".format(value))

       print "————————————————————————————————————————"
       print("white: D{} (A{}V)".format(white_level,white_volts))

       # Wait before repeating loop
       time.sleep(delay)
       except KeyboardInterrupt:
pass
```

7.3 PCB Designing

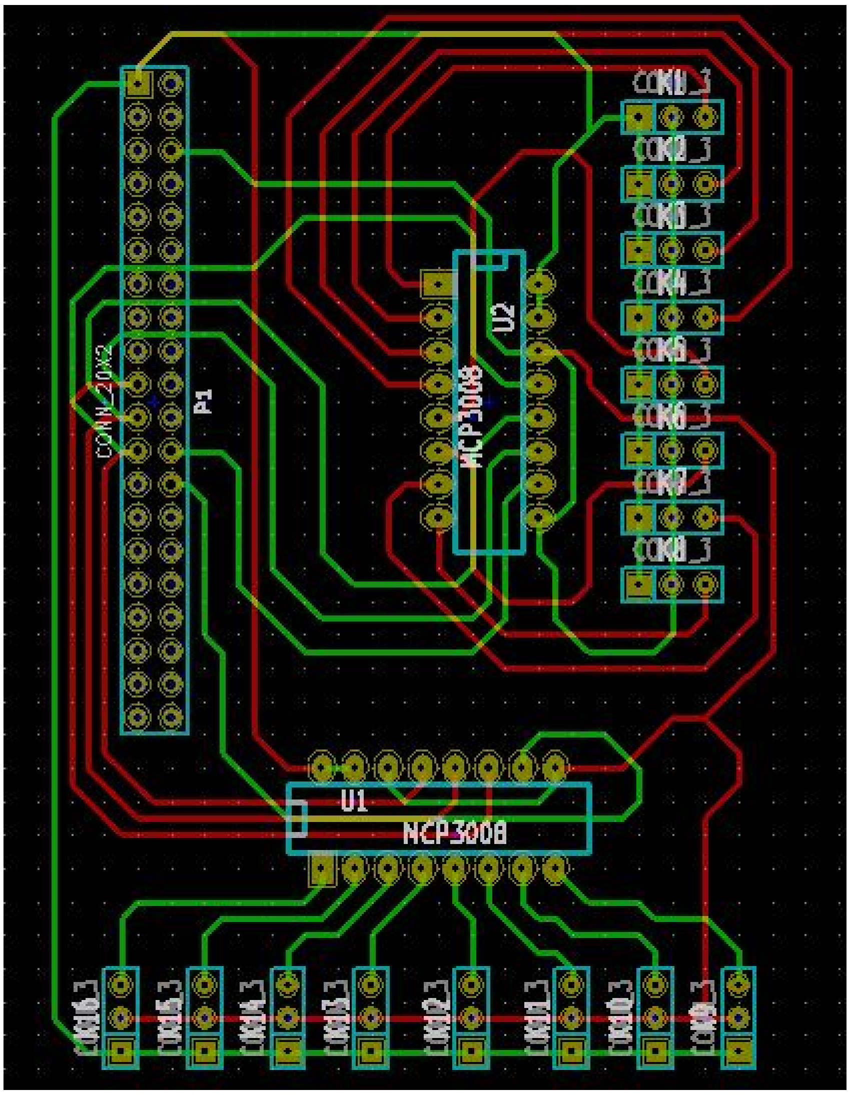

Figure 6: PCB

8 References

1. www.raspberrypi-spy.co.ukanalogue-sensors-on-the-raspberry-pi-using-an-mcp3008/

2. www.en.wikipedia.org/wiki/SerialPeripheralInterface

3. www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/

4. developers.google.com/edu/python/

5. www.alldatasheet.com/Mcp3008+datasheet