Toy Car Control Using Esp32 and Mobile & Cordic Algorithm

Guided by

Mr.T.Naresh Sir,
Assistant professor,M.Tech,
Department of Electronics and Communication Engineering,
RGUKT,RK-Valley

Submitted by

R170212-A.Gowri Priya

Introduction to Esp32

- Toy car can be controlled by using Vamon board in which Esp32,Arm and Fpga are fabricated.
- ESP32 can interface with other systems to provide Wi-Fi and Bluetooth functionality through its SPI / SDIO or I2C / UART interfaces.
- ESP32 is highly-integrated with in-built antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules.
- ESP32 adds priceless functionality and versatility to your applications with minimal Printed Circuit Board (PCB) requirements.
- ESP32 is capable of functioning reliably in industrial environments, with an operating temperature ranging from –40°C to +125°C. Powered by advanced calibration circuitries, ESP32 can dynamically remove external circuit imperfections and adapt to changes in external conditions.

Toy Car Control Using Esp32 and Mobile

Components Required

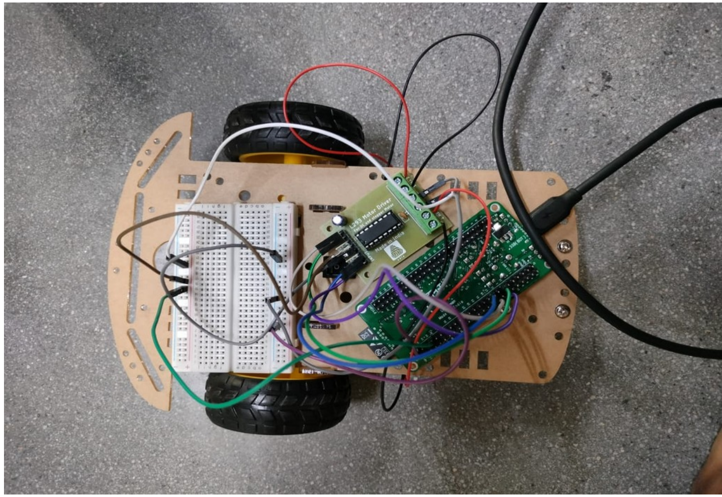
1) Esp32 Deveopment Kit
2) Vamon Board
3) Toy Car with Motors
4) L293D Motor Driver IC
5) Dabble app(Download from playstore)
6) Breadboard
7) Connecting Wires

Circuit Connections

- Refer the pin diagram of L293D motor driver IC.

- Connect Vcc1 pin to Vin pin on Esp32.
- The input and enable pins(ENA, IN1, IN2, IN3,IN4 and ENB) of the L293D IC are connected to six Esp32 digital output pins(14, 16, 17, 18,19 and 15).
- Connect one motor to across OUT1 & OUT2 and the other motor across OUT3 & OUT4.
- Connect external 5V to the Vcc2 pin of L293D motor driver IC.
- Connect external GND pin to the GND pin of L293D motor driver IC
- Go to ardunio IDE and Write the code.

- Click on Compile and Upload the code to ”DOIT ESP32 DEVKITV1”(Vamon board) by using UART.
- Now open Dabble app search for bluetooth devices and Select ”MyEsp32”and connect it.
- After connecting click on GamePad Icon and you will get control panel.
- In this way we can control Toy car by using Esp32 and Mobile.



Toy Car

- CORDIC:- COordinate Rotation DIgital Computer

- In general we have Mathematical Formulas to find trigonometric,hyperbolic and exponential functions:

1) $\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots\ldots$

2) $\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots\ldots$

3) $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots\ldots$

4) $\cosh(x) = \dfrac{e^{x} + e^{-x}}{2}$

The above functions can be implemented by taylor series with some approximations.It includes multiplication of x with x.

1011*1011=

1011

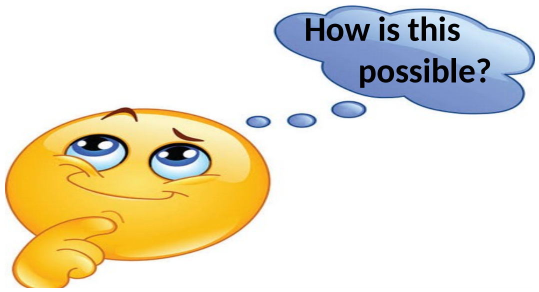
1011+

0000+

1011+

1111001

- A simple multiplication needs many adders,which in turn increase hardware.
- We want a multiplierless hardware with greater speed and lower power consumption.



- This can be possible by using Cordic Algorithm.

Cordic Algorithm

- Cordic Algorithm was proposed by Volder in 1959.It is improved by Walther in 1971.
- It is widely used hardware efficient iterative algorithm for computation of elementary functions such as hyperbolic,arithematic,trigonometric,exponential, logorithmic functions.

Basic concept of this algorithm is

- To decompose the desired rotation angle into sum of set of predefined elementary rotation angles(lookup table).
- Such that rotation through each of them can be accomplished with simple shift and add operations,which reduces hardware cost.
- For this Cordic algoritm we require only
  *3 adders
  *1 look up table (register to store predefined values)
  *2 shift registers

- Cordic algorithm generally works by rotating the coordinate system through a constant set of angles until the angle is reduced to zero.

- For example,to track 60 degrees

  60=45+26.6-14+7 $\longrightarrow$ These are predefined angles(α_i)

$$Z_{i+1} = Z_i - d_i \alpha_i$$

- when $z_{i+1} = 0$,it tracks the angle 60 degrees.

$$d_i = \begin{cases} +1, z+ve \\ -1, otherwise \end{cases}$$

- $$Z_{i+1} = Z_i - d_i\alpha_i$$

= 60-45=15 (+ve)

= 15-26.6=-11 (-ve)

=-11+14=3 (+ve

= .

=.

$$d_i = \begin{cases} +1, z+ve \\ -1, otherwise \end{cases}$$

till $z_{i+1} = 0$ to track required angle

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

$$x_{i+1} = \cos\phi(x_i - y_i \tan\phi)$$

$$y_{i+1} = \cos\phi(y_i + x_i \tan\phi)$$

$$Let \quad \tan\phi = \pm 2^{-i}$$

$$A_n = \frac{1}{\cos\phi} = \sec\phi = \sqrt{1 + \tan^2\phi} = \sqrt{1 + 2^{-2i}}$$

- Where A_n is the scaling factor to reduce the equation

- For n iterations

$$x_n = A_n(x_0 \cos z_0 - y_o \sin z_o)$$

$$y_n = A_n(y_0 \cos z_0 + x_0 \sin z_0)$$

$$z_n = 0$$

$$A_n = \prod \sqrt{1 + 2^{-2i}} = \sqrt{1} \times \sqrt{1.5} \times \sqrt{1.25} \times \sqrt{1.0625} \times \ldots..$$

$$A_n = 1.6476$$

- Then we take

$$x_0 = \frac{1}{A_n} = 0.6073$$

$$y_0 = 0$$

$$z_0 = someangle$$

$$Then$$

$$x_n = \cos z_0$$

$$y_n = \sin z_0$$

Look up table

- **LOOK UP TABLE:**

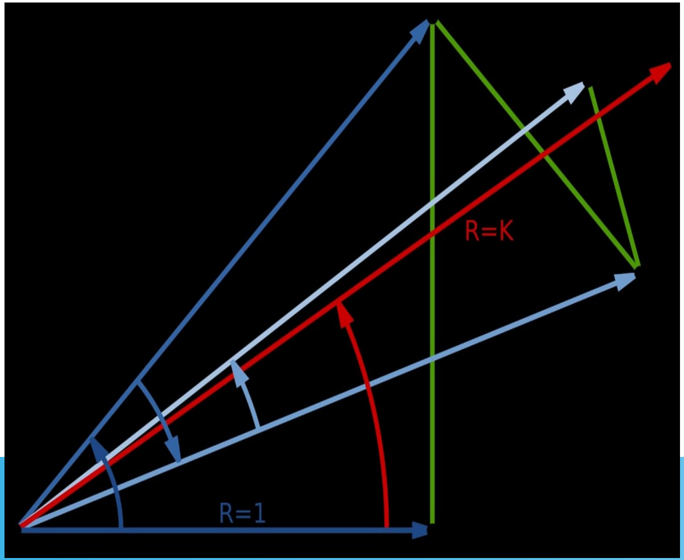
$$\tan\phi = 2^{-i}$$

$$\phi = \tan^{-1} 2^{-i}$$

0	45
1	26.6
2	14
3	7.1
4	3.6
5	1.8
6	0.9
7	0.4
8	0.2
9	0.1

$$\textit{for example to calculate } \sin 60^0, \cos 60^0,$$

$$60 = 45 + 26.6 = 71.6 - 14 = 57.6 + 7.1 = 64.7 - 3.6$$
$$= 61.1 - 1.8 = 59.3 + 0.9 = 60.02 - 0.4 = 59.8 + 0.2$$
$$= 60.$$



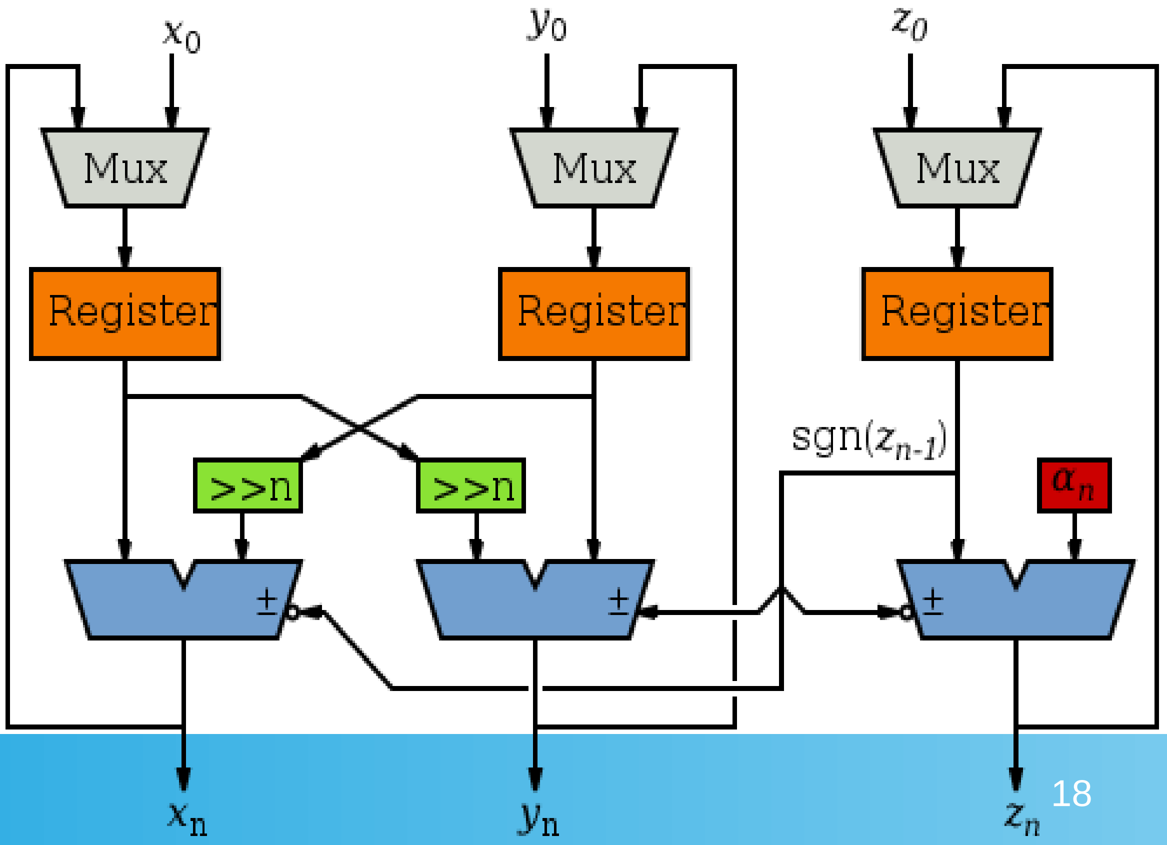
ARCHITECTURE OF CORDIC ALGORITHM

$$x_{i+1} = x_i - d_i 2^{-i} y_i$$

$$y_{i+1} = y_i + d_i 2^{-i} x_i$$

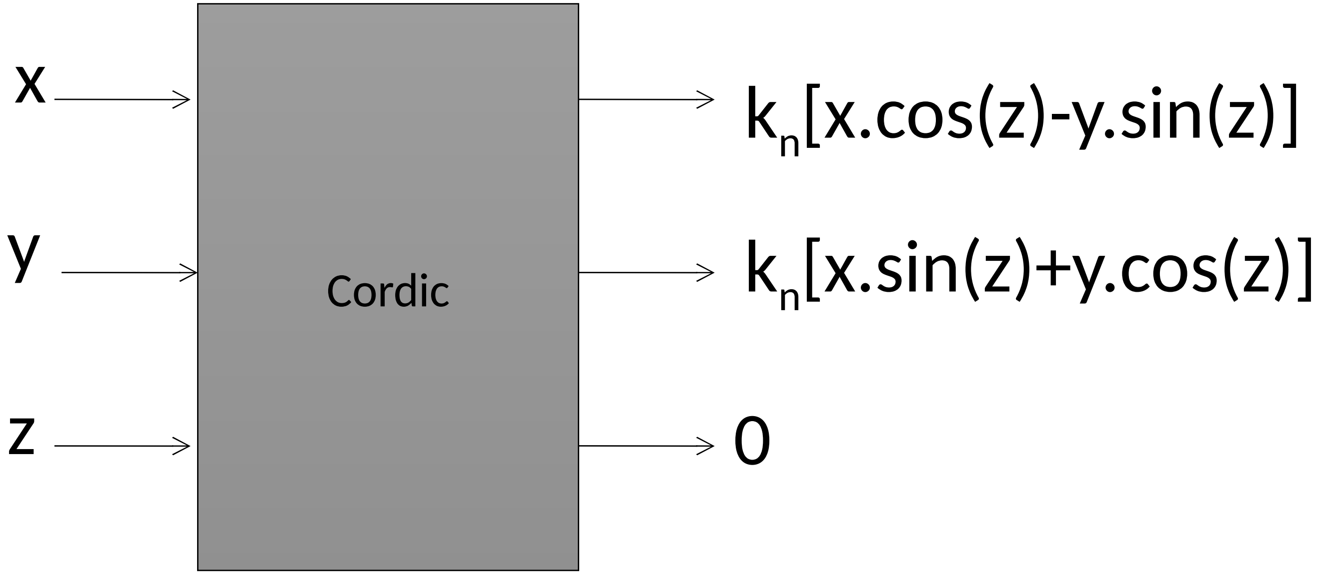
$$z_{i+1} = z_i - d_i \alpha_i$$

$$\alpha_i = \tan^{-1}(2^{-i})$$



Implementation of sine and cosine by using cordic algorithm

- **Implementing cos(θ) and sin(θ)**



We are using 32 bit in our program.If bit size increases precision increases.

45	00101101000000000000000000000000	2d000000
26.6	00011010100100001010001111000000	1a90a3c0
14	00001110000010010011011101001011	0e09374b
7.1	00000111001000000000000000000000	07200000
3.6	00000011100100110111101001011110	039374bc
1.8	00000001110010100011110101110000	01ca3d70
0.9	00000000110110011001100110011001	00d99999
0.4	00000000011100101011000000100000	0072b020
0.2	0000000000111001010110000001000	00395810
0.1	00000000000111001010110000001000	001cac08
0.6073	10011011011110000000000000011	09b7800

Python code for trigonometric functions

```
from __future__ import division

from math import atan,pi,sqrt

import matplotlib.pyplot as plt

# Calculate the arc Tan table once

ArcTanTable = []

for i in range(50):

 ArcTanTable.append(    atan( 2.0**(-1 * i) )    )

# Calculate the scaling factor K once

KN = []

value = 1.0

for i in range(50):

        value = value * sqrt(  1.0 + 2.0**(-2 * i)  )

        KN.append(1.0 / value)
```

```
s=[]
c=[]
t=[]
co=[]
secant=[]
cosec=[]
iterations=50
count=0
def tan():
	for i in range(0,361):
		if i==90 or i==270:
			#t.append(None)
			continue
		else:
			t.append(s[i]/c[i])
			secant.append(1/c[i])
```

def cot():

for i in range(0,361):

if i==0 or i==180 or i==360:

#t.append(None)

continue

else:

co.append(c[i]/s[i])

cosec.append(1/s[i])

- #calculating amplitudes of sin and cos in first quadrant
- for j in range(0,361):
  - count=j

```
if j>270:
    j=j-360
elif j>90:
    j=j-180
beta = j* pi / 180.0
Vx,Vy = 1.0 , 0.0
for i in range(iterations):
    if beta < 0:
        Vx,Vy = Vx + Vy * 2.0**(-1 * i)  ,  Vy - Vx * 2.0**(-1 * i)
        beta = beta + ArcTanTable[i]
    else:
        Vx,Vy = Vx - Vy * 2.0**(-1 * i)  ,  Vy + Vx * 2.0**(-1 * i)
        beta = beta - ArcTanTable[i]
Vx,Vy = Vx * KN[iterations - 1] , Vy * KN[iterations - 1] #Vx=cos amplitude ,#Vy=sin amplitude
```

```
        if len(s)>90 and len(s)<=270: #for 2nd and 3rd quadrants
            s.append(-Vy)
            c.append(-Vx)
        else:                   #for first and fourth quadrants
            s.append(Vy)
            c.append(Vx)
    tan()
    cot()
    ###########   sin and cos ############
```

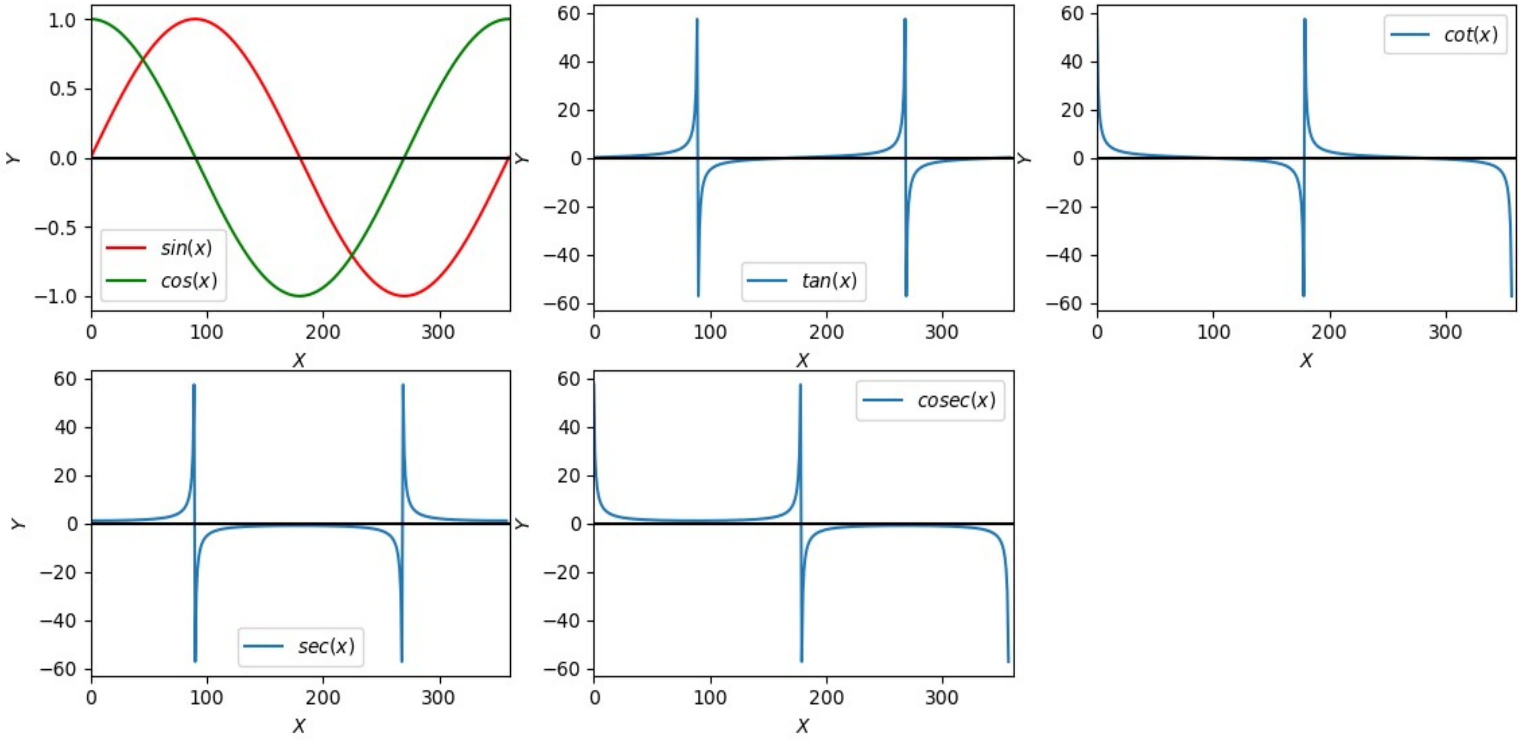
- plt.subplot(2,3,1)
- plt.xlim([0,361])
- plt.plot(s,'r',label='$sin(x)$') #sin
- plt.plot(c,'g',label='$cos(x)$') #cos

```
plt.axhline(y = 0, color = 'k', linestyle = '-')
plt.xlabel('$X$')
plt.ylabel('$Y$')
plt.legend()

###########   tan #############
plt.subplot(2,3,2)
plt.xlim([0,361])
plt.plot(t,label='$tan(x)$')#tan
plt.xlabel('$X$')
plt.ylabel('$Y$')
plt.legend()
plt.axhline(y = 0, color = 'k', linestyle = '-')
```

```
############## cot    #############
plt.subplt.plot(co,label='$cot(x)$')#tan
plt.xlabel('$X$')
plt.ylabel('$Y$')
plt.legend()
plt.axhline(y = 0, color = 'k', linestyle = '-')

############## secant    #############
plt.subplot(2,3,4)
plt.xlim([0,361])
plt.plot(secant,label='$sec(x)$')#tan
plt.xlabel('$X$')
plt.ylabel('$Y$')
plt.legend()
plt.axhline(y = 0, color = 'k', linestyle = '-')
```

```
############## cosec    #############
plt.subplot(2,3,5)
plt.xlim([0,361])
plt.plot(cosec,label='$cosec(x)$')#tan
plt.xlabel('$X$')
plt.ylabel('$Y$')
plt.legend()
plt.axhline(y = 0, color = 'k', linestyle = '-')
plt.show()
```

Simulation result



Uses of cordic algorithm

Cordic algorithm used for the calculations of

- Trigonometric functions
- Inverse trigonometric functions
- square roots
- Hyperolic functions
- Division and multiplication
- Logorithmic functions
- Exponential functions

Applications of cordic algorithm

Apllications of cordic algorithm are

- DSP(for MAC blocks)
- Image processing
- 3D Graphics
- Robotics
- Digital filters
- Solving linear systems


Thank You