CONTROL PC VOLUME USING ROTARY ENCODER
Domain: "INTERNET OF THINGS"

A MINI PROJECT REPORT

Submitted by

SOBHIKA.R	**113220031138**
NANDHINI.K	**113220031080**
VINUNITHI.GN	**113220031167**

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

VELAMMAL ENGINEERING COLLEGE
[An Autonomous Institution]

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2022

BONAFIDE CERTIFICATE

Certified that this Mini project report "**CONTROL PC VOLUME USING ROTARY ENCODER"** is the bonafide work of **SOBHIKA R (113220031138), NANDHINI K (113220031080)** and **VINUNITHI GN(113220031167)** who carried out the Mini project work under my supervision.

SIGNATURE **SIGNATURE**

MRS. DR P.PRITTO PAUL

SUPERVISOR

Computer Science and Engineering

Velammal Engineering College

Ambattur – Red hills Road,
Chennai – 600066.

DR.B.MURUGESHWARI

HEAD OF THE DEPARTMENT

Computer Science and Engineering,

Velammal Engineering College,

Ambattur–Red hills Road,
Chennai – 600066.

MINI PROJECT EXAMINATION

The MINI PROJECT Examination of this project work "**CONTROL PC VOLUME USING ROTARY ENCODER"** is a bonafide record of project done at the Department of Computer Science and Engineering, Velammal Engineering College during the academic year 2021 – 2022 by

SOBHIKA.R	**113220031138**
NANDHINI.K	**113220031080**
VINUNITHI.GN	**113220031167**

of Second year Bachelor of Engineering in Computer Science and Engineering submitted for the university examination held on_____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

A rotary encoder is a type of position sensor that converts the angular position (rotation) of a knob into an output signal that is used to determine what direction the knob is being rotated. Due to their robustness and fine digital control; they are used in many applications including robotics, CNC machines and printers. One of the key areas which need to be looked at while developing such systems is the code implementation stage. In order to manage the work we shall be using C for the implementation of the code. We feel that if we successfully meet our goals then we shall have contributed towards the future of natural gesture based interfaces, if only in a minimal way.

Since rotary encoders are basic components, they do not come pre-programmed and need to be programmed manually to perform the desired actions. You'll also need to set up a build environment that will support your rotary encoder. A pre-built environment can be found here. Once you've set up your build environment, you can now program your rotary encoder to perform whatever action you need. This will require a bit of programming experience, but if you're just looking to perform some basic functions like volume control and scrolling, the code snippets for some of those actions can be found here.

INDEX

CHAPTER 1

INTRODUCTION

In order to control pc using rotary encoder which enables us to control certain functions on our computer/Laptop. A rotary encoder detects rotation of the center shaft and is used to control machinery position and motor speed, audio volume, the cursor position on an LCD (Liquid Crystal Display) screen, or simply LED brightness.

A rotary encoder, also called a shaft encoder, is an electro-mechanical device that converts the angular position or motion of a shaft or axle to analog or digital output signals. There are two main types of rotary encoder: absolute and incremental. The output of an absolute encoder indicates the current shaft position, making it an angle transducer. The output of an incremental encoder provides information about the motion of the shaft, which typically is processed elsewhere into information such as position, speed, and distance**.**

For example, in Chapter 1 (Internet radio), the radio station and volume were selected by turning a rotary encoder. The incremental rotary encoder has 20 positions, and the rotor is continuously rotated clockwise or anti-clockwise to increase or decrease a control variable. The rotary encoder has two pins, termed A or CLK (clock) and B or DT (data), and a common pin. As the rotary encoder rotor is turned, pins A and B each make contact with the common pin, which generates square waves, but as the pins are offset, the square waves are 90° out of phase. The number of square wave pulses indicates the extent of the rotation, which is measured on either pin A or pin B.

The internet of things helps people live and work smarter, as well as gain complete control over their lives. In addition to offering smart devices to automate homes, IOT is essential to business. IOT provides businesses with a real-time look into how their systems really work, delivering insights into everything from the performance of machines to supply chain and logistics operations.

Increasingly, organizations in a variety of industries are using IOT to operate more efficiently, better understand customers to deliver enhanced customer service, improve decision-making and increase the value of the business. An IOT network leverages a combination of mobile, cloud, and Big Data technologies along with data analytics and low-cost computing to enable the collection and exchange of data among physical objects connected within the network. And what's impressive is that all of this is accomplished with minimal human intervention. Industries use IOT for efficient management and optimizing production processes. In an exceptional time of technological miracles, IOT has given us Machine to machine communication. Many brands aim to and some like Microsoft have already capitalized on Internet of things.

We live in an exciting age of technological and digital revolution. In just a decade, we've witnessed a radical change in the world around us. Thanks to the recent advancements in Data Science, today, we have at our disposal things like AI-powered smart assistants, autonomous cars, surgical bots, intelligent cancer detection systems, and of course, the Internet of Things (IOT). The Internet of Things is a major sensation of the 21st century. After all, who would have thought that someday we'd have access to a technology that would allow us to connect everyday objects – like thermostats, kitchen appliances, door lock systems, baby monitors, and electrical appliances – over a centralized and integrated network and control them from anywhere in the world! . IOT isn't just limited to

everyday household objects – you can even connect sophisticated industrial objects and systems over an IOT network. As of now, there are over 7 billion IOT devices, and this number is expected to grow to 22 billion by 2025!. As the IOT technology continues to gain momentum in the modern industry, researchers and tech enthusiasts are readily investing in the development of pioneering IOT projects.

ARDUINO

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.

The Arduino platform has become quite popular with people just starting out with electronics, and for good reason. Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board -- you can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package.

The Arduino hardware and software was designed for artists, designers, hobbyists, hackers, newbies, and anyone interested in creating interactive objects or environments. Arduino can interact with buttons, LEDs, motors, speakers, GPS units, cameras, the internet, and even your smart-phone or your TV! This flexibility combined with the fact that the Arduino software is free, the hardware boards are pretty cheap, and both the software and hardware are easy to learn has led to a large community of users who have contributed code and released instructions for a variety of Arduino-based projects.

For everything from robots and a heating pad hand warming blanket to honest fortune-telling machines, and even a Dungeons and Dragons dice-throwing gauntlet, the Arduino can be used as the brains behind almost any electronics project.

Leonardo

The Leonardo differs from all preceding boards in that the ATmega32u4 has built-in USB communication, eliminating the need for a secondary processor. This allows the Leonardo to appear to a connected computer as a mouse and keyboard, in addition to a virtual (CDC) serial / COM port

.The Arduino Leonardo is a microcontroller board based on the ATmega32u4. It has 20 digital input/output pins (of which 7 can be used as PWM outputs and 12 as analog inputs), a 16 MHz crystal oscillator, a micro USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

ATmega32U4
The high Performance, low power AVR® 8-bit microcontroller.

Built-in USB communication
The ATmega32U4 has built-in USB communication that allows the Micro to appear as a mouse/keyboard on your machine.

Battery Connector
The Arduino Leonardo features a barrel plug connector, that works great with a standard 9V battery.

EEPROM
The ATmega32U4 features 1kb of EEPROM, a memory which is not erased when powered off.

Board	**Name**	Arduino® Leonardo
	SKU	A000057
Microcontroller	ATmega32u4	
USB connector	Micro USB (USB-B)	
Pins	**Built-in LED Pin**	13
	Digital I/O Pins	20
	Analog input pins	12
	PWM pins	7
Communication	**UART**	Yes
	I2C	Yes
	SPI	Yes

Power	**I/O Voltage**	5V
	Input voltage (nominal)	7-12V
	DC Current per I/O Pin	10 mA
	Power Supply Connector	Barrel Plug
Clock speed	**Processor**	ATmega32U4 16 MHz
Memory	**ATmega32U4**	2.5KB SRAM, 32KB FLASH, 1KB EEPROM
Dimensions	**Weight**	20 g
	Width	53.3 mm
	Length	68.6 mm

Rotary encoders

Rotary encoders deliver pulses, which you count, to get position feedback from the motion system. I have only used them on PLC-controlled systems for position feedback and machine timing, never with a stepper motor, but it essentially works like this: Condition is met to pulse motor to a certain point

There are two general types of rotary encoder. The ***absolute type***, which is more expensive, can track all of its previous positions. Basically, all positions of the knob on a rotary encoder have a value. The way that value is read depends on what sensor the encoder uses: *magnetic*, *optical*, or *mechanical*.

Another type of encoder is the ***incremental type***. Just like the absolute, the incremental encoder is further categorized according to sensor type. The electromechanical encoder consists of a rotating disk with electrical contacts spaced at 90 degrees. The electrical contacts are wired through the "+" pin while the rest of the space is connected to the "GND" pin.

There are several different types of rotary encoders. I'm restricting this discussion to the simpler "incremental" encoders. These are sometimes called *quadrature* or *relative* rotary encoders.
These encoders have two sensors and output two sets of pulses. The sensors, which can be magnetic (hall effect) or light (LED or Laser), produce pulses when the encoder shaft is rotated.
As there are two sensors in two different positions they will both produce the same pulses, however, they will be out of phase as one sensor will pulse before the other one. Which sensor goes first is determined by the direction of rotation.

The encoder can be mounted exactly like a potentiometer, and it has a D-shaft to accept a knob. It also has its own push button momentary contact switch that can be activated by pressing down upon the shaft.

The absolute rotary encoder is one that measures an absolute angle of the encoded shaft through having a unique code for each shaft position. With that, every position of the measurement range/angle is being identified by a certain code on a disc. This means negates the need for counters as positional values are always directly available even when power is removed from the encoder.

A mechanical absolute encoder is a common low-cost option that is constructed with a metal disc and works as follow:

- A metal disc on a shaft is used in conjunction with a stationary pickup device and rotates
- When the shaft rotates, a unique code pattern is produced
- Which means each position of the shaft has a pattern
- This pattern is used to determine the exact position

The above explanation applies to how a mechanical absolute rotary sensor works, but there are two other ways to detect rotational position changes; Optical or magnetic sensors changes.

The pinouts of the control encoder are as follows:
GND – The Ground connection.
+V – The VCC or positive supply voltage, usually 3.3 or 5-volts.
SW – The pushbutton switch output. When the shaft is depressed this goes to ground level.
DT – The Output A connection.
CLK – The output B connection.

Motor encoders are mounted on the shaft of a DC motor and can count the amount that the motor shaft has moved. Coupled with a motor driver, this will allow you to create a feedback mechanism that can permit you to specify how much you would like the shaft to turn.
You can also use these encoders as a tachometer to measure the speed that the shaft is rotating.

1.1 Objective/Aim of the Project :

Implemented a simple Arduino and rotary encoder to control pc volume.

1.2 Existing Problem Identified:

How to control your computer using encoders?

1.3 Proposed Solution:

By using Arduino sensor **and rotary encoder that converts the physical turns into a digital signal to your computer**. When you install this onto your keyboard, you can map the rotational data to something like volume control, scrolling on your computer screen

1.4 Proposed Work:

This paper introduces a technique based on determining distance by the sensor and accordingly a particular function is performed. This project uses the HID library with a rotary encoder switch as a digital volume control for PC sound. The push button connected to the encoder shaft is used as a mute/unmute button. One side of the switch has two pins , that is the push switch. The side with three contacts is the rotary part of the switch. Circuit power comes from the USB connection.

CHAPTER 2

SYSTEM MODEL

2.1 Principle Behind Our Proposed Scheme

The Rotary encoders are simple components that convert a physical turn of the knob into a digital signal. Twisting a rotary encoder either clockwise or counterclockwise converts to a digital signal that is sent from the rotary encoder to the computer to perform any programmed action. These actions typically require a repeated button press, making a rotary encoder ideal for performing actions like volume adjustment, scrolling through a webpage, tabs, or windows, and mass undoing or redoing actions. Without a rotary encoder, these actions take significantly more time but using one makes performing these actions much faster and easier.

2.2 Circuit Diagram

The circuit diagram of Arduino part of the model is shown in the below figure.

- A USB HID microcontroller –ARDUINO BOARD.
- A rotary encoder (Here, a stepper motor is being used).
- A suitable USB data cable for the microcontroller board.

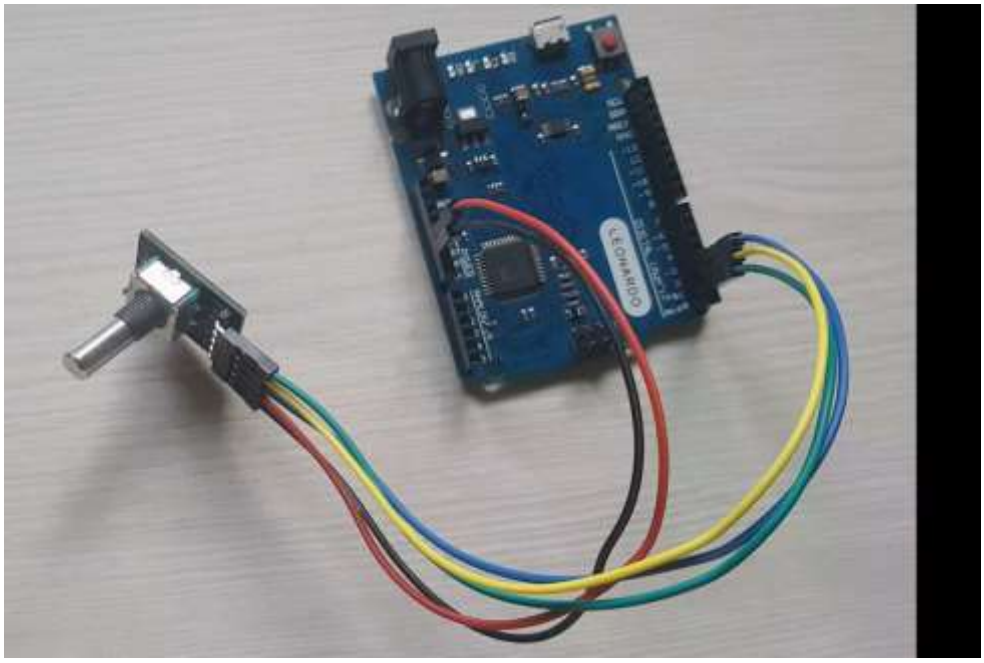


Figure2.2 The circuit diagram of Arduino part of the model.

2.3 Problem Formulation

In this paper, we have implemented a simple Arduino with rotary encoder controlled computer where we can control few functions like: Press Windows +R to open the Run window and type in shell :startup. Start playing music on the computer. As you turn the knob, the volume should go up one way, down the other. Push down on the switch and the sound should be muted, another push, unmute.

2.4 Architecture / Design of the Proposed Solution

The circuit diagram below is a simple demonstration of how to use Rotary Encoder with Arduino. Assemble the same circuit on breadboard or PCB.

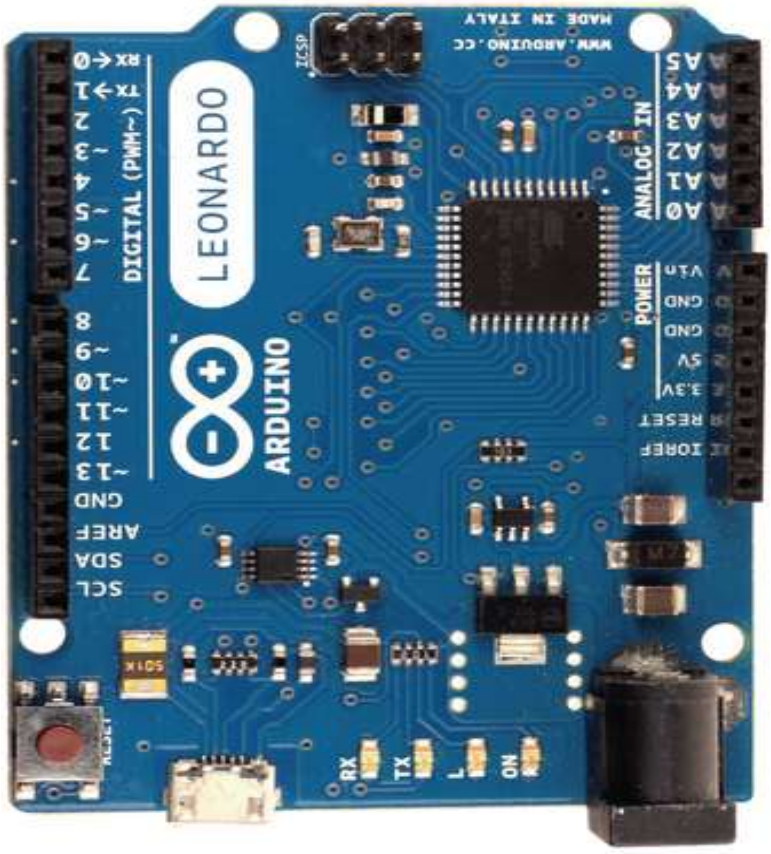
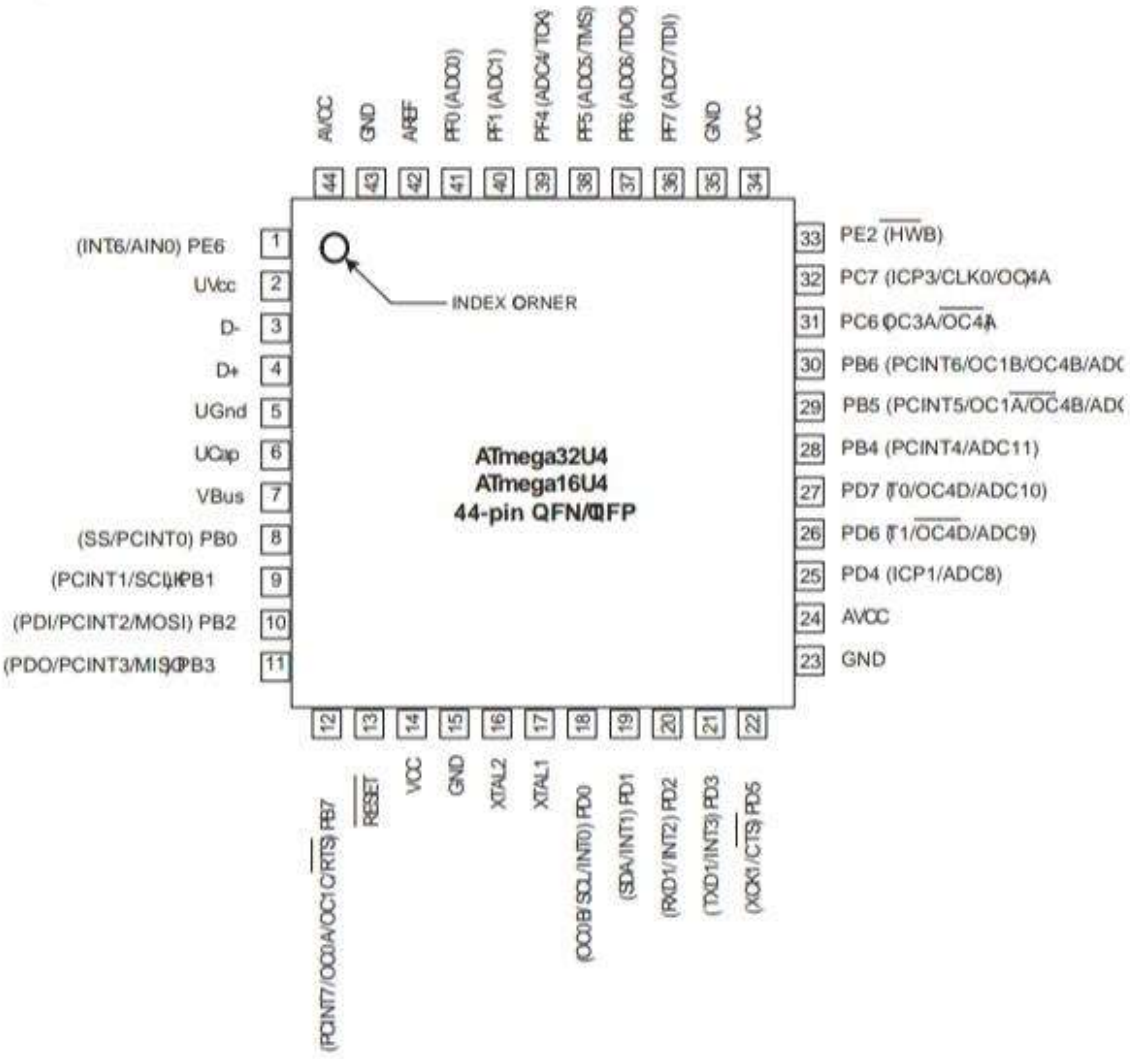


Figure 2.3:Simple demostration of how to use to rotary encoder with arduino

The ATmega16U4/ATmega32U4 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the device achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

1. Pin Configurations

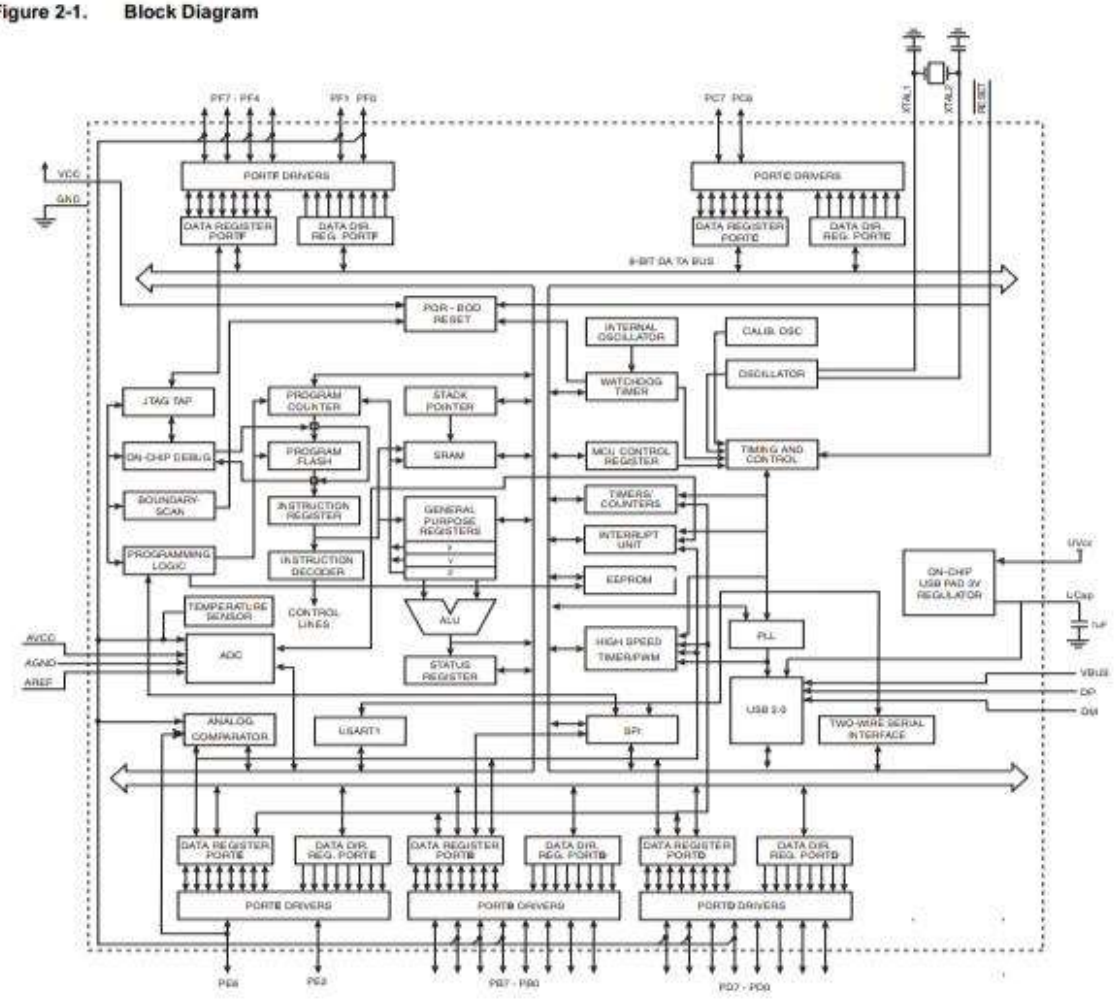
Figure 1-1. Pinout



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers. The device provides the following features: 16/32K bytes of In-System Programmable Flash with Read-WhileWrite capabilities, 512Bytes/1K bytes EEPROM, 1.25/2.5K bytes SRAM, 26 general purpose I/O lines (CMOS outputs and LVTTL inputs), 32 general purpose working registers, four flexible Timer/Counters with compare modes and PWM, one more high-speed Timer/Counter with compare modes and PLL adjustable source, one USART (including

CTS/RTS flow control signals), a byte oriented 2-wire Serial Interface, a 12-channels 10-bit ADC with optional differential input stage with programmable gain, an on-chip calibrated temperature sensor, a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, IEEE std. 1149.1 compliant JTAG test interface, also used for accessing the On-chip Debug system and programming and six software selectable.

Figure 2-1. Block Diagram



CHAPTER 3

IMPLEMENTATION

A. Components Required

- Arduino UNO x 1
- Rotary module with push switch
- USB Cable (for Arduino)
- Few Connecting Wires
- A Laptop with internet connection

B. Detailed Modeling

The design of the circuit is too simple, but the setup of the components is very significant.
An encoder can be installed directly on the motor shaft or made as a module. The rotary encoder module, including 5 pins, is the most common rotating encoder. 2 pins support encoder supply, SW is a push button on the module, and CLK and DT show the A and B channels. Some of the features of this module are:

- The ability of Rotate to infinity
- 20 pulse resolution
- 5V supply voltage

The knob which we rotate is internally connected to the disk(Pic – 2). If we rotate clockwise or counter-clockwise it will move accordingly. The grey portion is GND and Golden contact points are connected to Vcc. There are two contact point placed at a specific distance apart which are nothing but our CLOCK and DATA line.

As we rotate our encoder, the two output will change depending upon the position of the encoder. Which will generate two trains of pulses.

If you look closely those two signals will be 90 degrees out of the phase. If the encoder rotates clockwise then CLOCK will lead and if the encoder rotates counter-clockwise then DATA will lead.

And taking a look at state change for clockwise two signals will have opposite values and for counter-clockwise same values.

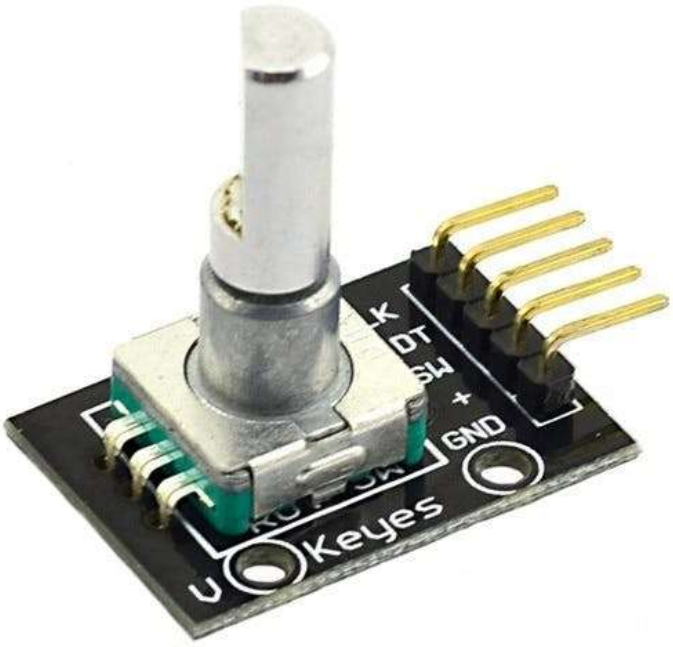


Figure3.1 The rotary encoder with five pins – Clock, Data, Switch, Vcc, GND

To use a rotary encoder, we should count the pulses of channels A and B. To do this, we used Arduino UNO and performed three projects for positioning the encoder, controlling the LED light and controlling the speed and direction of the DC motor.

Rotary Encoder Module Pinout

This module has 5 pins:

- **CLK:** Output A
- **DT:** Output B
- **SW:** Push Switch output (It is normally High, by pressing the knob, it will be LOW).
- **VCC:** Module power supply – 5V
- **GND:** Ground

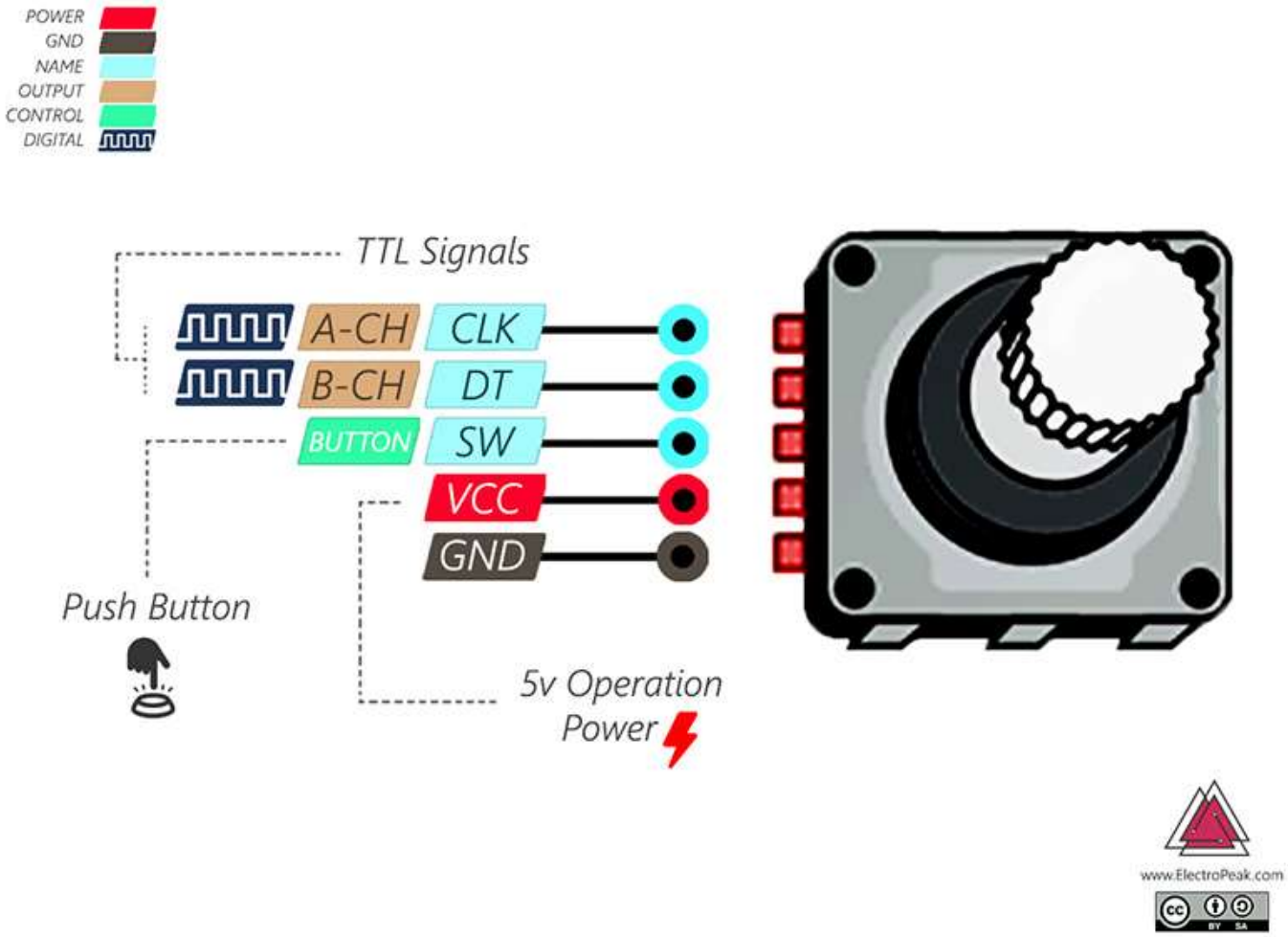


Figure 3.2: Most common rotating encoder. 2 pins support encoder supply, SW is a push button on the module, and CLK and DT show the A and B channels.

A rotary incremental encoder has two output signals, A and B, issuing square waves when the encoder shaft rotates. The square wave frequency indicates the speed of shaft rotation, while the A-B phase relationship indicates the direction of rotation.
Whereas some rotary incremental encoders determine the amount of rotation through a separate counter that counts the number of pulses outputted in response to the amount of rotational displacement of the shaft. Such rotary incremental encoders go through such process of determining position with magnetic rotary encoders:

- Rotary displacement of the shaft occurs
- Resulting from that, the encoder outputs a pulse string accordingly
- A separate counter placed at the reference point then counts the number of pulses to produce the desired output

Connect the + to 5V, GND to GND pin, CLK to pin number 6, and DT to pin number 7.
You need to know the position of the shaft to use the encoder. The position of the shaft varies depending on the amount of its rotation. It changes from 0 to infinity for clockwise rotation, and from 0 to minus infinity for the counter clockwise rotation.

Step 1: Arduino Sketch & Hardware Assembly

- Firstly, we'll connect the Arduino with the Rotary Encoder using some Jumper cables.

- The table below shows the digital pins of Arduino I used to connect it to the encoder:

Figure 3.3 The Arduino IDE is connected to PC.

- Next, Connect the Arduino IDE to the PC and also Download the Arduino IDE.
- In the Arduino IDE, Goto Tools > Port and make sure an appropriate port is selected. In my case it was COM11. Next, Goto Tools > Board and select the type of Arduino board that you have, in my case it was Arduino Leonardo.
- Goto Sketch > Include library > Manage libraries. In the window that opens up, search for 'HID-Project' and install the HID-project library by Nicohood.

- Next, open this link to open the GitHub page of this project and copy the code from there to the Arduino IDE.
- Currently, in the repository, you will find the code to increase or decrease the volume. For that, I have used the Nicohood library that we installed earlier. I have used *Consumer.write(MEDIA_VOLUME_UP)* to increase the volume. Similarly,

- MEDIA_VOLUME_DOWN to decrease the volume.

Step 2: Configuring different modes

- The knob of the rotary encoder can also be pushed like a button. We can read the value of the button from the *SW* pin of the encoder.

  So let's define different modes for our encoder. Mode 1 will increase and decrease the volume, Mode 2 will change the tabs of the browser, Mode 3 will control Light room sliders and PowerPoint presentations and Mode 4 will increase or decrease the size of the brush in photo shop.

  So in our code, let's declare 2 variables called *mode* and *max Modes*. The *mode* variable will store the currently selected mode, which by default would be mode 1. While the max Modes variable will store the total number of modes we want to configure, which in my case would be 4.

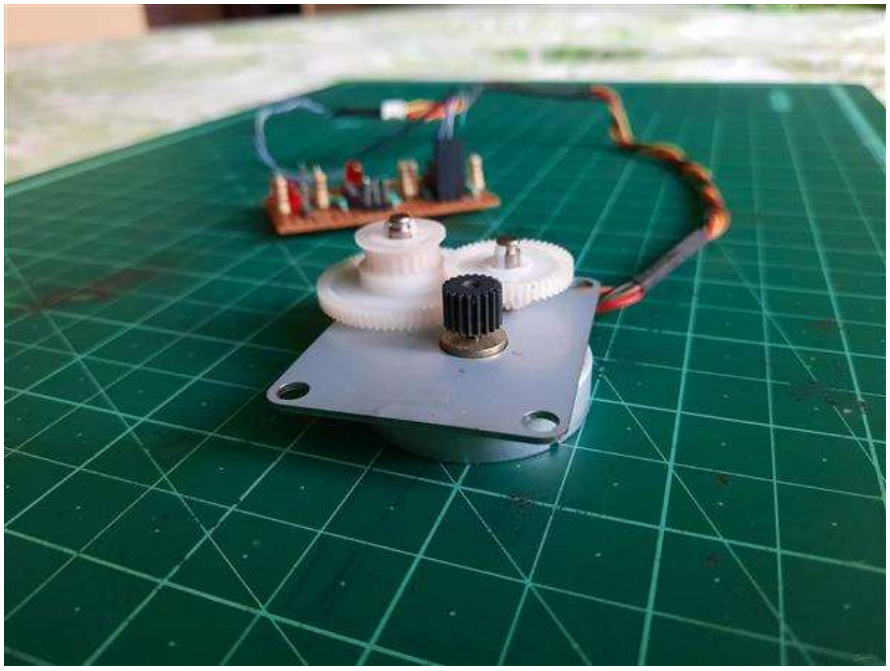


Figure 3.4 the rotary encoder with knob

Figure 3.5. Digital outputs of the encoder to the headers A0 and A1 of the Arduino board.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation. Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction. Program Flash memory space is divided in two sections, the Boot Program section and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The SPM instruction that writes into the Application Flash memory section must reside in the Boot Program section. During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the Reset routine (before subroutines or interrupts are executed). The Stack Pointer (SP) is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture. The memory spaces in the AVR architecture are all linear and regular memory maps. A flexible interrupt module has its control registers in the I/O space with an additional Global Interrupt Enable bit in the Status Register. All interrupts have a separate Interrupt Vector in the Interrupt Vector table. The interrupts have priority in accordance with their Interrupt Vector position. The lower the Interrupt Vector address, the higher the priority.

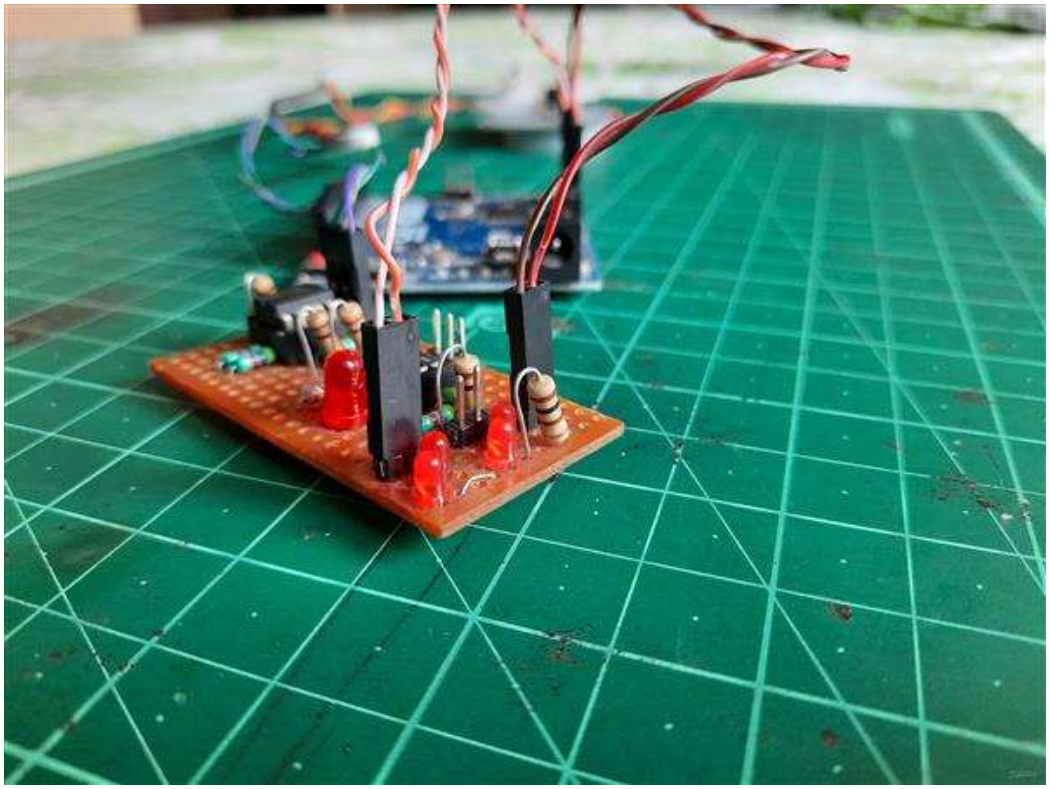


Figure 3.6 Connect the digital outputs of the encoder to the headers A0 and A1 of the Arduino board in any order

Connect the +ve of the input power of the encoder to the +5-volt header of the Arduino board and the -ve of the encoder to the GND header. A separate power supply for the servo motor. I always recommend doing this as the servo can induce electrical noise onto the 5-volt line that the Arduino uses. But, if you really must, you can eliminate the extra supply and use the Arduino 5-volt output.
Connect the digital outputs of the encoder to the headers A0 and A1 of the Arduino board in any order. If you swap the order of the output wires, the direction of the encoder registered by the microcontroller will get reversed.
Make sure the Volume.exe program is running and check it by pressing the buttons on the keyboard as shown in the video. After that, check if rotating the encoder in either direction changes the volume.

C. Working Principle

The encoder has a disk with evenly spaced contact zones that are connected to the common pin C and two other separate contact pins A and B, as illustrated below.

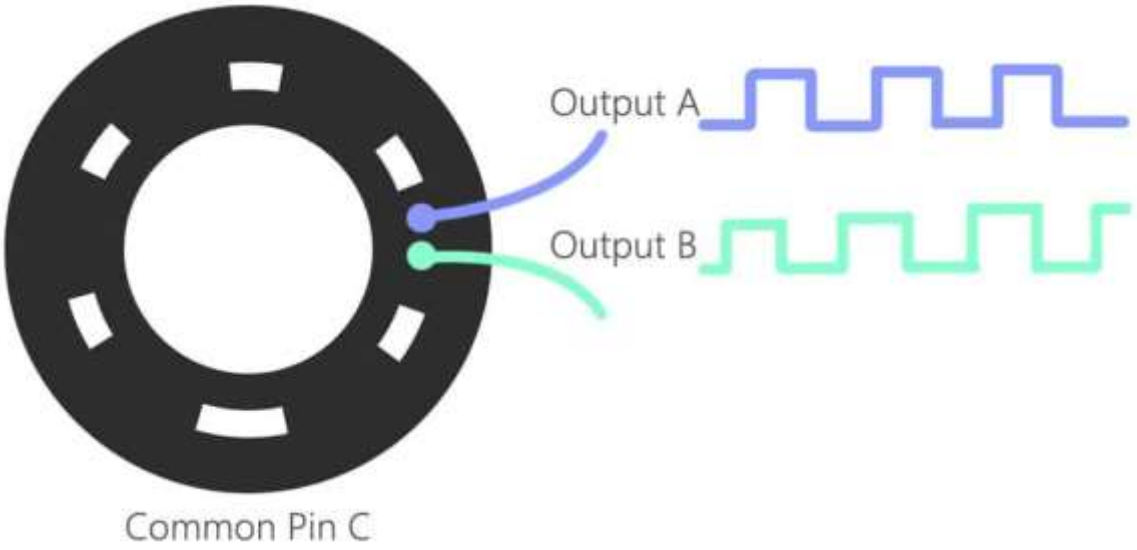


Figure 3.7 The input and output wavelength from the pin c.

When the disk will start rotating step by step, pins A and B will start making contact with the common pin and the two square wave output signals will be generated accordingly.

Any of the two outputs can be used for determining the rotated position if we just count the pulses of the signal. However, if we want to determine the rotation direction as well, we need to consider both signals at the same time.

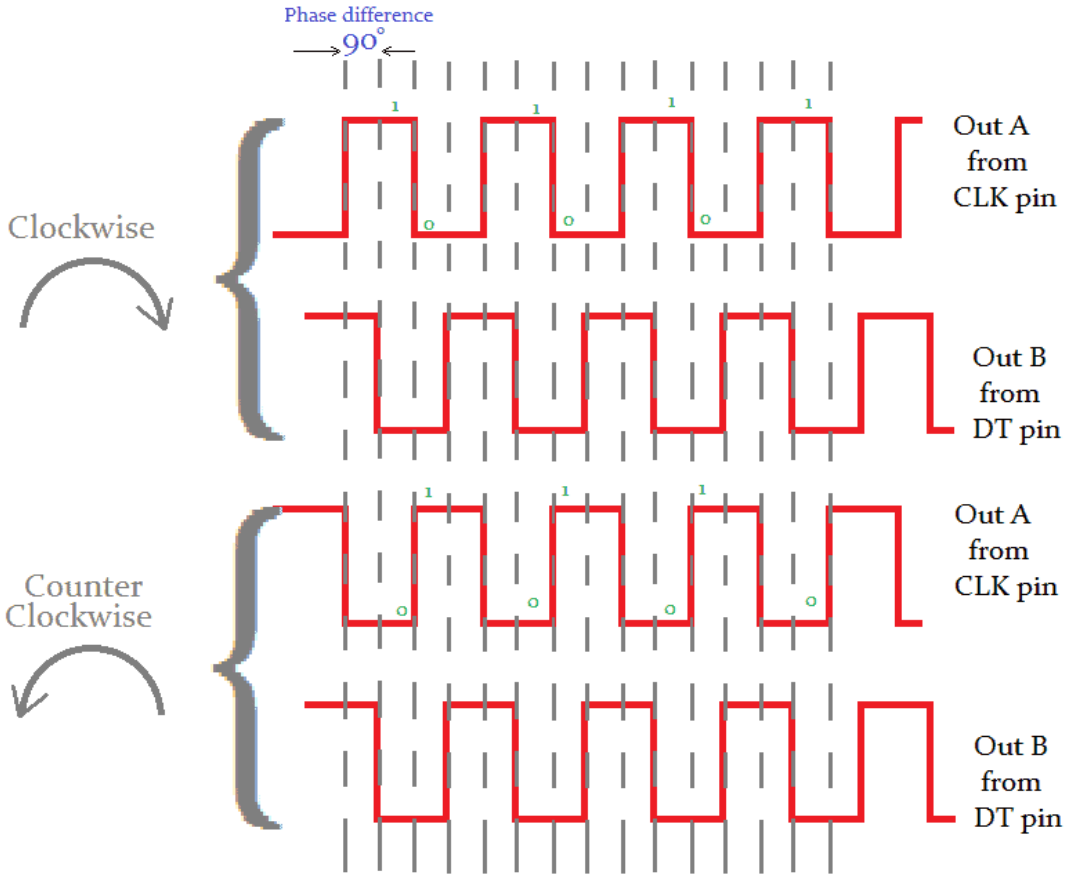


Figure 3.8 The phase difference of the rotary encoder is shown clockwise and anti-clockwise.

We can notice that the two output signals are displaced at 90 degrees out of phase from each other. If the encoder is rotating clockwise the output A will be ahead of output B.

So if we count the steps each time the signal changes, from High to Low or from Low to High, we can notice at that time the two output signals have opposite values. Vice versa, if the encoder is rotating counter-clockwise, the output signals have equal values. So considering this, we can easily program our controller to read the encoder position and the rotation direction.

- A method called change Mode which will increment the currently selected mode when the button of rotary encoder is pressed.

Here, when this method is called, it will increase the mode by 1. When the mode is equal to max Modes and this method is called again, the currently selected mode will jump back to 1.

- In Case 1, we are simply increase the volume as we did earlier.
- In Case 2, we want to cycle between the browser tabs. We can cycle the browser tabs using the key combination Control + Tab. So in the code as well, I have sent Control + tab as well. To configure that, I have used the arduino Keyboard library. You can find all the key definitions used here and more in the official documentation of keyboard library.

- In case 3, to control the sliders of light room and also to change the slides of a PowerPoint presentation, we can use the up/down arrow keys. Similarly, in case 4, to increase the size of the brush we can user the shortcut to increase brush size i.e. by pressing.

We start the sketch by defining some constants to represent the inputs from the encoder and the outputs to the two LEDs.
Next, a few integer variables are defined. The counter variable represents the count that will be modified by turning the controller, it can be a positive or negative number..
The current State CLK and previous State CLK variables hold the state of the CLK output (Output B), these are used as part of the system to determine the direction of rotation.
A string called encdir is defined, it will be used when we print the current direction of rotation to the serial monitor.

Now on the the Setup section.

The Setup is pretty straightforward, we setup the serial monitor, define the connections to the encoder as inputs and the connections to the two LEDs as outputs.
At the end of the Setup we read the current value of the CLK pin and assign it to the previous State CLK variable.
And now the Loop, where all the action takes place!

We check the CLK input and compare it to the previous State CLK value. If it has changed then a pulse has occurred

Next the logic to determine the direction of rotation. We now look at the DT pin on the encoder module and compare it to the current state of the CLK pin.

If it is different then we are rotating counter clockwise. We then decrement the counter variable value, set encdir to hold a value of “CCW” and turn on the red (CCW) LED.
If, on the other hand, the two values are the same then we are moving clockwise. We do the opposite – increment the counter, write “CW” to the encdir variable and turn on the Green (CW) LED.
After we print our results to the serial monitor we update previous State CLK with the current state of CLK. Then we do it all over again.

Explanation of the code:

In the beginning of the sketch, we declare variables to store the readings from the CLK pin (clkPin), DT pin (dtPin), and SW pin (switchPin). Next we declare a variable called count to store the count number. The count will start at zero, so it’s initially set equal to zero. The clkPinLast variable keeps track of the last state of the CLK pin. It’s initially set equal to LOW. Then we declare another variable called clkPinCurrent that will store the current state of the CLK pin and set it equal to LOW as well.
In the setup() section, we use pin mode to set each pin as an INPUT. We don’t want the switchPin to float, so we use the internal pullup resistor. We will print the counts to the serial monitor so we initialize that too.

The first part of the loop() section is used to reset the count to zero when the rotary encoder switch is pressed. To do that, we first create a local variable called switchState, and set it equal to the digital read of the switchPin. The if statement on the next line sets count equal to zero when the switchPin goes LOW.

The first part of the loop() section is used to reset the count to zero when the rotary encoder switch is pressed. To do that, we first create a local variable called switchState, and set it equal to the digital read of the switchPin. The if statement on the next line sets count equal to zero when the switchPin goes LOW.
If the knob is not turning counter clockwise, then it must be turning clockwise. So we can use an else statement to increase the count variable by one if the above if statements are not true.

If the knob is not turning counter clockwise, then it must be turning clockwise. So we can use an else statement to increase the count variable by one if the above if statements are not true.

USING INTERRUPTS WITH ROTARY ENCODERS

If you want to use the rotary encoder to control another device, the Arduino will need to balance taking reads from the encoder with sending signals to the other device. Unfortunately, the Arduino isn’t very good at doing multiple things at the same time.

For example, say you want to have an LED blinking on and off at the same time the encoder is counting up and down. Blinking an LED uses the delay() function, which causes the Arduino to stop what it's doing for the duration of the delay. During that time it won't be able to detect clicks from the rotary encoder.

This sketch is very similar to the sketch in the previous project, with a couple differences. Since now we need to trigger interrupt service routines, we have to connect the clkPin and dtPin to pins 2 and 3 instead of pins 3 and 4 as in the last sketch.

We also need to add interrupt service routines for the clkPin and dtPin. Before variables can be used in interrupt service routines, they need to be declared as volatile. In this sketch count, clkPinLast, and clkPinCurrent are used in the interrupt service routine so they are declared as volatile variables. We also need to declare a variable for the LED pin (ledPin).

Next we create the interrupt service routine with void encoderInterrupt(). The code that goes inside the interrupt is the same code from the loop() section in the previous example that reads the dtPin and clkPin and increments and decrements the counter.

In the setup() section, we set the pin modes for each pin. The clkPin, dtPin, and switchPin are set the same as in the last example. But this time we also have an LED, so the ledPin variable is set as an OUTPUT.

The interrupt service routine needs to be called in the setup() section. To call the interrupt service routine, use the attachInterrupt() function.

The attachInterrupt() function takes three parameters – the digital pin to interrupt, the name of the interrupt service routine, and the mode of the interrupt. Each interrupt needs it's own attachInterrupt() function, so in this sketch there is one for pin 2 and one for pin 3

IV. RESULT AND DISCUSSION

Rotary encoders are pretty versatile, they can be used as both controls and as sensors.
As controls, they have much greater precision than a potentiometer, and they cost just a little bit more.
The will be building a complete motor controller using the rotary encoder included with my gear motor. Being able to position my robot to within less than a millimeter is an extremely attractive proposition.
Hope this article, and its associated video, have opened your eyes to some tasks you can perform with a rotary encoder.

V. APPLICATION

- These are used where speed, direction, acceleration, and monitoring rotation rate are required.
- These are used in different industries like material handling, packaging & conveyor.
- In the automation field, these encoders are utilized as sensors for speed, angle, acceleration & position.
- These are used to measure linear motion by using gear racks, spindles, cable pulls, or measuring wheels.
- These encoders are used to change a mechanical input to electrical signals by using tachometers, counters, PLC systems & PCs in industries.
- These are used in packaging, assembly machines, indication systems, printers, CNC machines, testing machines, robotics, textiles, motor feedback, medical equipment, drilling, and labelling machines.
- Automotive optical sensors
- Accurate position sensor for encoder
- Sensor for motion, speed, and direction
- Sensor for “turn and push” encoding

VI. CONCLUSION AND FUTURE SCOPE

According to this latest study, the 2021 growth of Solid Shaft Rotary Encoders will have significant change from previous year. By the most conservative estimates of global Solid Shaft Rotary Encoders market size (most likely outcome) will be a year-over-year revenue Impressive growth rate of in 2021, from USD million in 2020. Over the next five years the Solid Shaft Rotary Encoders market will register a Impressive CAGR in terms of revenue, the global market size will reach USD million by 2026.

Rotary Encoders are sensors that detect position and speed by converting rotational mechanical displacements into electrical signals and processing those signals. Sensors that detect mechanical displacement for straight lines are referred to as Linear Encoders. Rotary Encoders are used to control the speed and direction of electric motors. They can control multiple axes of motion. A shaft encoder is mounted directly to the motor for this application. These are used in packaging and labeling machines, textile manufacturing machinery or the lift industry.

SOURCE CODE

```
#include <HID-Project.h>
// Rotary Encoder Inputs
#define inputCLK 1
#define inputDT 2
#define inputSW 3
int currentStateCLK;
int previousStateCLK;
void setup() {
	// Set encoder pins as inputs
	pinMode (inputCLK,INPUT);
	pinMode (inputDT,INPUT);
	pinMode(inputSW, INPUT_PULLUP);
	// Setup Serial Monitor
	Serial.begin (9600);
	Consumer.begin();
	Keyboard.begin();
	// Read the initial state of inputCLK
	// Assign to previousStateCLK variable
	previousStateCLK = digitalRead(inputCLK);
}
void loop() {
	// Read the current state of inputCLK
	currentStateCLK = digitalRead(inputCLK);
	// If the previous and the current state of the inputCLK are different then a pulse has occured
	if (currentStateCLK != previousStateCLK){
		// If the inputDT state is different than the inputCLK state then
		// the encoder is rotating counterclockwise
		if (digitalRead(inputDT) != currentStateCLK) {
			rotateRight();
		} else {
			rotateLeft();
		}
```

```
}
        // Update previousStateCLK with the current state



previousStateCLK = currentStateCLK;
}
void rotateRight() {
        // Increase the volume.
        Consumer.write(MEDIA_VOLUME_UP);
}
void rotateLeft() {
        // Decrease the volume.
        Consumer.write(MEDIA_VOLUME_DOWN);
}
```