An Incremental Optical Encoder has two channels (A and B) positioned 90° out of phase. These channels can be used together to indicate angular position and direction, as well as calculate angular velocity.
Any transducer that generates a coded reading of a measurement can be termed an **encoder**.

**Shaft Encoders** are **digital transducers** that are used for measuring angular displacements and velocities.

**Relative advantages** of digital transducers over their analog counterparts:

- High resolution (depending on the word size of the encoder output and the number of pulses per revolution of the encoder)
- High accuracy (particularly due to noise immunity of digital signals and superior construction)
- Relative ease of adaptation in digital control systems (because transducer output is digital) with associated reduction in system cost and improvement of system reliability
Shaft Encoders can be classified into **two categories** depending on the nature and method of interpretation of the output: **Incremental & Absolute Encoders**

**Incremental Encoders**
- Output is a pulse signal that is generated when the transducer disk rotates as a result of the motion that is being measured.
- By counting pulses or by timing the pulse width using a clock signal, both angular displacement and angular velocity can be determined.
- Displacement can be obtained with respect to some reference point on the disk, as indicated by a reference pulse (index pulse) generated at that location on the disk. The index pulse count determines the number of full revolutions.
Displacement can also be obtained by homing the system to define the zero position at a certain point in the actual machine or mechanism.

**Absolute Encoders**

- An absolute encoder has many pulse tracks on its transducer disk. When the disk of an absolute encoder rotates, several pulse trains – equal in number to the tracks on the disk – are generated simultaneously.

- At a given instant, the magnitude of each pulse signal will have one of two signal levels (i.e., a binary state) as determined by a level detector. This signal level corresponds to a binary digit (0 or 1). Hence, the set of pulse trains gives an encoded binary number at any instant.
– The pulse windows on the tracks can be organized into some pattern (code) so that each of these binary numbers corresponds to the angular position of the encoder disk at the time when the particular binary number is detected.
– Pulse voltage can be made compatible with some form of digital logic (e.g., TTL)
– Direct digital readout of an angular position is possible.
– Absolute encoders are commonly used to measure fractions of a revolution. Absolute encoders can also have an additional track that generates an index pulse for every revolution, as in the case of an incremental encoder. This index pulse can be used in the homing procedure which defines a reference position for the motor, as well as another way to measure complete motor revolutions.
In Binary Code, bit switching may not take place simultaneously.

**Schematic Diagram of an Absolute Encoder Disk Pattern**
(a) Binary code
(b) Gray code

Ambiguities in bit switching can be avoided by using gray code. However, additional logic is needed to covert the gray-coded number to a corresponding binary number.

Absolute Encoders must be powered and monitored only when a reading is taken. Also, if a reading is missed, it will not affect the next reading.
Elements of the Optical Encoder

- The optical encoder uses an opaque disk (code disk) that has one or more circular tracks, with some arrangement of identical transparent windows (slits) in each track.
- A parallel beam of light (e.g., from a set of light-emitting diodes) is projected to all tracks from one side of the disk.
- The transmitted light is picked off using a bank of photosensors on the other side of the disk that typically has one sensor for each track.
- The light sensor could be a silicon photodiode, a phototransistor, or a photovoltaic cell.
- Since the light from the source is interrupted by the opaque areas of the track, the output signal from the probe is a series of voltage pulses. This signal can be interpreted to obtain the angular position and angular velocity of the disk.

- Note that an incremental encoder disk requires only one primary track that has equally spaced and identical window (pick-off) areas. The window area is equal to the area of the inter-window gap. Usually, a reference track that has just one window is also present in order to generate a pulse (known as the index pulse) to initiate pulse counting for angular position measurement and to detect complete revolutions.
– In contrast, absolute encoder disks have several rows of tracks, equal in number to the bit size of the output data word. Furthermore, the track windows are not equally spaced but are arranged in a specific pattern on each track so as to obtain a binary code or a gray code for the output data from the transducer.

– It follows that absolute encoders need as least as many signal pick-off sensors as there are tracks, whereas incremental encoders need one pick-off sensor to detect the magnitude of rotation and an additional sensor at a quarter-pitch separation (pitch = center-to-center distance between adjacent windows) to identify the direction of rotation, i.e., the offset sensor configuration.
– Some designs of incremental encoders have two identical tracks, one a quarter-pitch offset from the other, and the two pick-off sensors are placed radially without any circumferential offset, i.e., the offset track configuration.

– A pick-off sensor for a reference pulse is also used.
<table>
<thead>
<tr>
<th>Wire</th>
<th>Function</th>
<th>Color</th>
<th>Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GND</td>
<td>Black</td>
<td>GND</td>
</tr>
<tr>
<td>2</td>
<td>Index</td>
<td>Green</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>CH A</td>
<td>Yellow</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Vcc</td>
<td>Red</td>
<td>5V</td>
</tr>
<tr>
<td>5</td>
<td>CH B</td>
<td>Blue</td>
<td></td>
</tr>
</tbody>
</table>
Decoding with Arduino

- The Arduino is an open-source platform with easy-to-use hardware and software.
- The board used in this project is the Arduino Duemilanove based on the microcontroller ATmega328.
- The objective is to measure angular position and angular velocity from an incremental optical encoder wired directly to the Arduino board.
The following two selectable decoding methods to measure angular position are implemented:

- **X2 Decoding for Two Incremental Encoders**

  - This code can measure angular position of two incremental encoders in X2 mode.
  - This means that the rising and falling edges of channel A of each encoder are watched via the interrupts of the Arduino.
  - Thus, a 500 PPR (pulses per revolution) encoder, for example, yields 1000 counts per revolution, which is twice the PPR of the encoder.
X2 Decoding

Forward Direction
A ≠ B

A = 1  A = 0  A = 1  A = 0  A = 1  A = 0
B = 0  B = 1  B = 0  B = 1  B = 0  B = 1

Reverse Direction
A = B

A = 1  A = 0  A = 1  A = 0  A = 1  A = 0
B = 1  B = 0  B = 1  B = 0  B = 1  B = 0
Wiring for X2 Decoding for Two Encoders

Optical Encoders

A. da Silva & K. Craig
- **X4 Decoding for One Incremental Encoder**
  - This code can measure the angular position of one incremental encoder in X4 mode.
  - This means that the rising and falling edges of channels A and B of the encoder are watched via the interrupts of the Arduino.
  - Thus, a 500 PPR encoder, for example, yields 2000 counts per revolution, which is four times the PPR of the encoder.
  - Both approaches measure angular velocity in counts per second. Angular position and angular velocity are then represented as PWM outputs.
  - A low-pass analog filter is used to convert the PWM signal of the analog outputs into an analog signal (0 to 5V).
X4 Decoding

Forward Direction

A=B
\[
\begin{align*}
A &= 1 \\
B &= 1
\end{align*}
\]
Transition in A

A\neq B
\[
\begin{align*}
A &= 1 \\
B &= 0
\end{align*}
\]
Transition in A

\[
\begin{align*}
A &= 0 \\
B &= 1
\end{align*}
\]
Transition in B

Reverse Direction

A=B
\[
\begin{align*}
A &= 1 \\
B &= 1
\end{align*}
\]
Transition in A

A\neq B
\[
\begin{align*}
A &= 1 \\
B &= 0
\end{align*}
\]
Transition in A

\[
\begin{align*}
A &= 0 \\
B &= 1
\end{align*}
\]
Transition in B

\[
\begin{align*}
A &= 0 \\
B &= 0
\end{align*}
\]
Transition in B

\[
\begin{align*}
A &= 1 \\
B &= 0
\end{align*}
\]
Transition in B

\[
\begin{align*}
A &= 0 \\
B &= 1
\end{align*}
\]
Transition in B

Optical Encoders

A. da Silva & K. Craig
Wiring for X4 Decoding for a Single Encoder
• The algorithm for X4 decoding is as follows:

- **When Transition in A occurs:**
  
  If $A \neq B$, Forward Direction, then increment position
  
  $\text{Pos} = \text{Pos} + 1$
  
  Else
  
  Reverse Direction, then decrement Position
  
  $\text{Pos} = \text{Pos} - 1$

- **When Transition in B occurs:**
  
  If $A = B$, Forward Direction, then increment position
  
  $\text{Pos} = \text{Pos} + 1$
  
  Else
  
  Reverse Direction, then decrement position
  
  $\text{Pos} = \text{Pos} - 1$
• **Angular Velocity Computation**
  – Angular velocity is computed as the derivative of position as follows:
    \[ v = \frac{dp}{dt} \approx \frac{p_i - p_{i-1}}{t_i - t_{i-1}} \]
  – \( p_{i-1} \) is position at time \( t_{i-1} \) and \( p_i \) is position at time \( t_i \).

• **Algorithm**
  – //Save current values of position and time
    – NewPositon = Pos, NewTime = ClockTime
  – //Calculate Velocity in Encoder Counts per Second
    – Vel = (NewPositon – OldPosition) / (NewTime – OldTime)
  – //Save current values of position and time for next calculation of velocity
    – OldPosition = NewPositon
    – OldTime = NewTime
  – //Time delay to run calculation of velocity again
    – Delay(1ms)
• The calculated velocity in encoder counts per second can be scaled and sent to the PWM output of the Arduino.

• The PWM outputs of the Arduino are 8 bit, thus the output varies from 0 to 255 ($2^8 = 256$).

• In order to measure positive and negative velocities, the mid-range of the output (127) is made equivalent to zero speed. Thus:

  $$
  \begin{align*}
  255 & \rightarrow \text{Maximum positive speed} \\
  127 & \rightarrow \text{Zero speed} \\
  0 & \rightarrow \text{Maximum negative speed}
  \end{align*}
  $$

• The PWM output of the Arduino is then calculated as follows:

  $$
  \text{AnalogOut} = 127 + 127 \frac{\text{Vel}}{\text{MotorMaxVel}}
  $$
• *Vel* can be positive or negative and vary from maximum positive motor velocity to maximum negative motor velocity.

• The *MotorMaxVel* in encoder counts per second is calculated as follows:

\[
MotorMaxVel = \left( MotorMaxVelRPS \right) \left( 4 \times EncoderPPR \right)
\]

• *MotorMaxVelRPS* is the maximum motor velocity in revolutions per second obtained from motor datasheet.

• *EncoderPPR* is the pulses per revolution of the encoder. The constant 4 represents the quadrature decoding.
Optical Encoders

MotorMaxVel = \left( \text{MotorMaxVelRPS} \right) \left( 4 \times \text{EncoderPPR} \right)

When Transition in A occurs:
- If A ≠ B, then change direction, then increment Position \( \rightarrow \text{Pos} = \text{Pos} + 1 \)
- Else, change direction, then decrement Position \( \rightarrow \text{Pos} = \text{Pos} - 1 \)

When Transition in B occurs:
- If A = B, then change direction, then increment Position \( \rightarrow \text{Pos} = \text{Pos} + 1 \)
- Else, change direction, then decrement Position \( \rightarrow \text{Pos} = \text{Pos} - 1 \)

//Save current values of position and time
\text{NewPosition} = \text{Pos}
\text{NewTime} = \text{ClockTime}

//Calculate Velocity in Encoder Counts per Second
\text{Vel} = \frac{\text{NewPosition} - \text{OldPosition}}{\text{NewTime} - \text{OldTime}}

//Save current values of position and time for next calculation of velocity
\text{OldPosition} = \text{NewPosition}
\text{OldTime} = \text{NewTime}

//Time delay to run calculation of velocity again
\text{Delay(1ms)}

\text{AnalogOut} = 127 + 127 \frac{\text{Vel}}{\text{MotorMaxVel}}
Wiring for X4 Decoding for a Single Encoder with Measurements of Position & Velocity

Angular Position
0 → 5V = 0 → 1 Rev

Angular Velocity
5V = 3000 RPM
2.5V = 0 RPM
0 V = -3000 RPM

Power Supply

Optical Encoders
• **Interrupts on the Arduino**
  – The frequency of the encoder channels A and B into the Arduino may be on the order of kHz.
  – The digital inputs of the Arduino are not fast enough to read these signals at this frequency.
  – Thus, interrupts need to be used. Arduino has two interrupts (0 and 1) that can be configured to trigger on a low value, a rising or falling edge, or a change in value.
  – The interrupts 0 and 1 are accessed on the following pins:
    • Interrupt 0 is on Digital Input 2
    • Interrupt 1 is on Digital Input 3
Thus, for the **X4 encoding** with a single encoder, the channels A and B are connected to pins 2 and 3 and the interrupts watch for transitions in both pins to count position up or down.

For the **X2 encoding** with two encoders, the channel A of one encoder is connected to pin 2 and the channel A of the second encoder is connected to pin 3. In the X2 mode only channel A is watched for transitions. When a transition occurs in channel A on either encoder, the state of the channel B (wired to an ordinary digital input) of the respective encoder is checked to properly identify the direction of the move and count the position up or down.
digitalReadFast(pin)

- The digitalReadFast(pin) command significantly improves the performance (by a factor of up to 50) to read the status of the channels A and B when an interrupt occurs.
- This command is not part of the default library of Arduino. To use this command with the open-source Arduino, the following steps are necessary:
  - Download the zip file dwf.zip from http://code.google.com/p/digitalwritefast/downloads/list
  - Put the digitalWriteFast folder into the library alongside the EEPROM, Ethernet, Firmata, etc. folders.
  - In the source code add: #include <digitalWriteFast.h>
  - Launch the Arduino environment. The digitalWriteFast will be inside the Sketch > Import Library menu.
  - The object code will not only be faster, but actually smaller as well.
• Open the Arduino Software
• Select File > Open… > select the pde file to be downloaded > click Open
- Select the correct board in Tools > Board
• Check the available ports before connecting the Arduino board to the USB port of your computer in Tools > Serial Port
• Connect the Arduino board to the USB port of the computer and select the new port that appears in Tools > Serial Port
• Click to Upload file.

// X4 Encoding for One Incremental Encoders

// Description:

// This code can measure angular position of one incremental en
channel A and B of the encoder is watched via two interru
• Watch the message bar.
• When it displays “Done uploading” the Arduino is ready to be used.