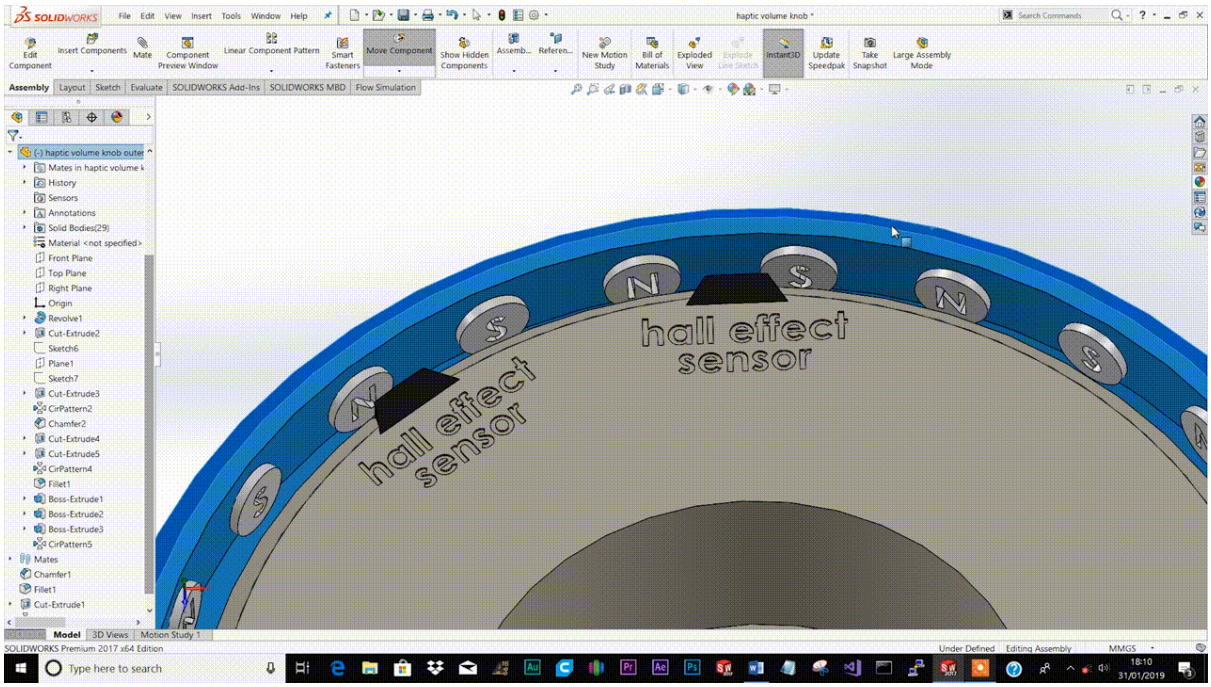
Magnetic rotary encoding

The method of detecting rotation that's discused on this page was born from my need for a rotary encoding solution that's contactless, continuous, bidirectional, cheap and easily customizable for different project types.

The goal of having a contactless rotation detection is mainly to alleviate the durability issues associated with traditional rotary encoding components like potentiometers and rotary encoders.

How it works.

The main idea here is to attach magnets with alternating poles to the rotating body, and then use two hall effect sensors to read the analog value changes that occurs when the body rotates. With this setup the circuit/sensors is completely isolated, and makes no physical contact with the rotating body.

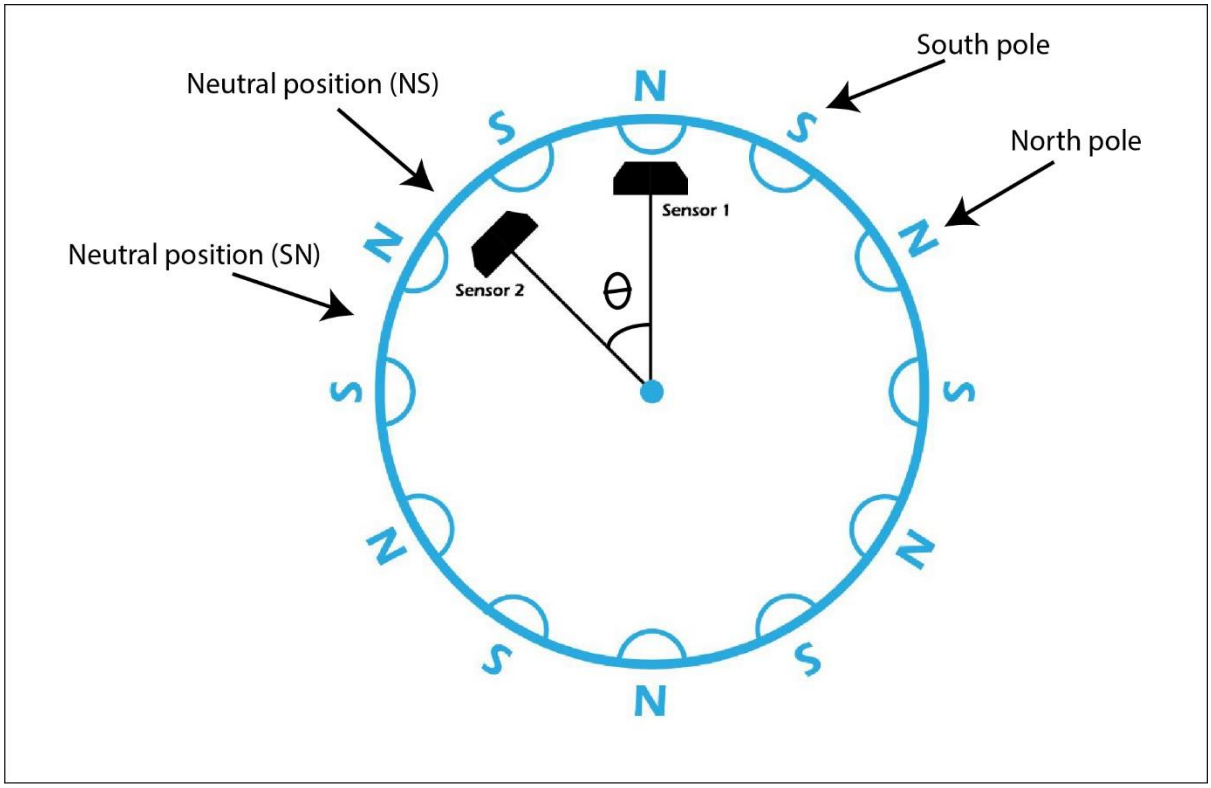


The animation above shows the basic operation of this setup, the outer ring with the alternating magnetic poles represents the moving body that revolves around two stationary hall effect sensors, a real life implementation is shown below, the magnets are glued to the outer ring, and the sensors are glued to the inner one…

The placement of the sensors is based on an Angle (θ), which can be calculated using the formula:

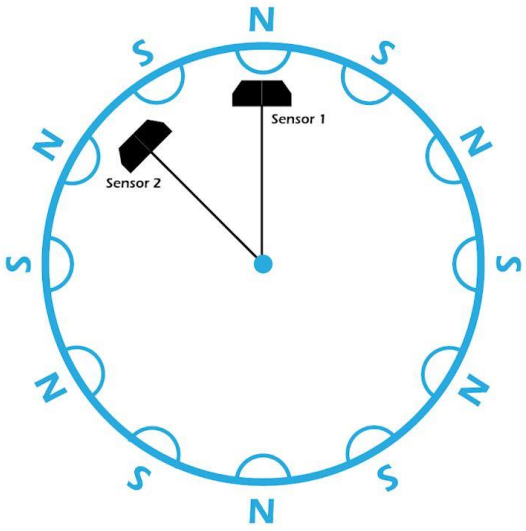
$$Angle(\theta) = \left(\frac{Number\ of\ magnets\ used(must\ always\ be\ even)}{360}\right) * (Steps\ apart + 0.5)$$

Where *"steps apart"* is a positive integer value that's less than half of the number of magnets used, this value determines how far apart the sensors will be from each other.



The angle (θ) offsets the sensors from each other in such a way that one of them will always be in the neutral position whenever the other is facing either of the poles. from this arrangement, the following tables can be derived.

The tables show the state of each sensor with respect to the other.. so when sensor 1 is facing north, sensor 2 will be in the neutral position, which is represented as a zero on the tables. from that information, one can further deduce the state that represents one clockwise step and one anticlockwise step, based on the four possible start positions…



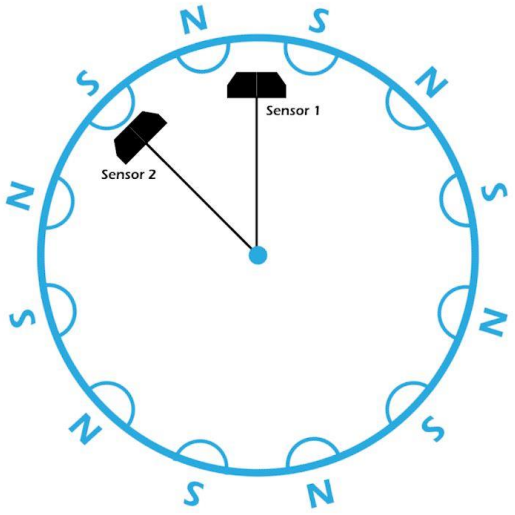
Sensor 1

	Start positions	Anti-clockwise step	clockwise step
N	N	NS	SN
	NS	S	N
S	S	SN	NS
	SN	N	S

Sensor 2

	Start positions	Anti-clockwise step	clockwise step
O	O	S	N
	S	SN	NS
O	O	N	S
	N	NS	SN

for example, if sensor 1 is currently in-between N and S, Its clear that if its state changes to S, the outer ring must have moved one step anticlockwise and if its state changes to N, it must have moved one step clockwise.



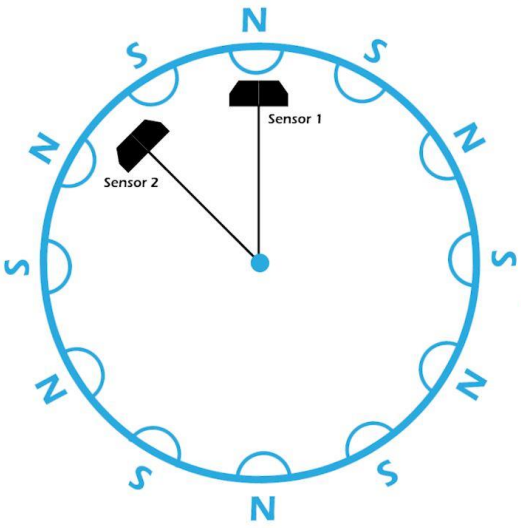
Sensor 1

	Start positions	Anti-clockwise step	clockwise step
N	N	NS	SN
	NS	S	N
S	S	SN	NS
	SN	N	S

Sensor 2

	Start positions	Anti-clockwise step	clockwise step
O	O	S	N
	S	SN	NS
O	O	N	S
	N	NS	SN

let’s say it did move clockwise to N, this becomes the new start position, and once again, one can tell that if its state changes to NS that’s an anticlockwise rotation and if it changes to SN, a clockwise rotation.



Sensor 1

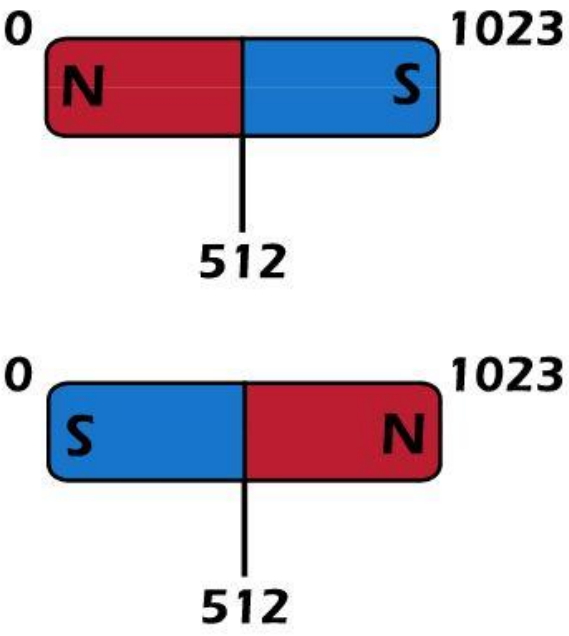
	Start positions	Anti-clockwise step	clockwise step
N	N	NS	SN
	NS	S	N
S	S	SN	NS
	SN	N	S

Sensor 2

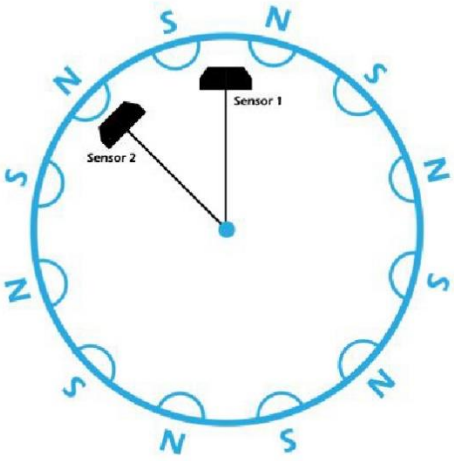
	Start positions	Anti-clockwise step	clockwise step
O	O	S	N
	S	SN	NS
O	O	N	S
	N	NS	SN

The continuous repetition of this process is how the rotations are registered.

Another thing worth mentioning is the operation of the hall effect sensors, the range of analog values from the sensor is divided into two, so the lower half could be assigned to the North pole and the upper half to the south, or vice versa.



what this means is the middle position will always be the same value regardless of the orientation of the magnet, so NS, or SN, its all the same to the sensor… therefore all the “NS” and “SN” on the tables are all synonymous to the neutral position, this means each table only accounts for 50% of the data required to accurately detect the rotations… which is why this setup requires two sensors.



Sensor 1

	Start positions	Anti-clockwise step	clockwise step
N	N	O	O
	O	S	N
S	S	O	O
	O	N	S

50%

100%

Sensor 2

	Start positions	Anti-clockwise step	clockwise step
O	O	S	N
	S	O	O
O	O	N	S
	N	O	O

50%

Please note that all the states indicated on the tables are representing a range of ADC values between 0 – 1023. For example:

- 0 – 400 represents the North pole of the magnet (N).
- 401 – 599 represents the Neutral state of the magnet (NS, SN and 0).
- 600 – 1023 represents the South pole of the magnet (S).

This values are pertinent to the rotation detection algorithm of the MagRotaryEncoder library.

MagRotaryEncoder Library.

This library was written based on the rotation detection method described above, so if you intend to use this method for detecting rotations in your projects, this library will come in handy. The library also features a haptics controller, which allows you to connect and control a vibration motor as a step count haptic feedback.

Using the library

```
#include <MagRotaryEncoding.h>


MagRotaryEncoder knob = MagRotaryEncoder(A0,A1); // create new encoding object and specify the Arduino analog pins connected to the hall effect sensors

int countsteps = 0;


void setup() {

  knob.set_poleStateValues(250,512,750); // set the peak ADC values for the (northpole, neutralstate, southpole)

  knob.set_haptics(6,50,255);  //set haptic feedback variables (arduino pwm pin, duration of haptics(ms), pwn strength from 0-255)
```

```
  knob.initialize_encoder();
  Serial.begin(9600);
}
void loop() {
countsteps += knob.detect_rotation();  // function returns a signed integer based on the number of rotation steps detected
  Serial.println(countsteps);
  knob.recaliberate_startPosition();
}
```

The code snippet above shows the basic use of the library.

Line 2 shows how to create a magnetic rotary encoding object, and also how you specify the analog pins on the Arduino that will be used to read the ADC values from the hall effect sensors.

The first line in the setup section is used to set the ADC values that represents the magnetic states discussed above.

The second line in the setup section is optional, only use this function when you have a haptics circuit setup.

In the main loop, the function detect_rotation() returns a signed integer “1” if a clockwise rotation is detected, “-1” if a counter clockwise rotation is detected and a “0” if no rotation is detected.

The function recaliberate_startPosition() does exactly what its name implies, this function should be called before or after the detect_rotation() function.