eTarget

1. **Executive Summary**

This document describes the circuit and algorithm used to create a low cost electronic target for general use.

2. **Applicable Documents**

This section contains documents that are relevant to the understanding of this project.

2.1. Web Sites

Law of Cosines

https://www.mathsisfun.com/algebra/trig-cosine-law.html

2.2. Technical References

2.3. Software

2.4. Glossary

Term	Description	Value
d	Distance between north/south and east/west sensors	100 mm
report time	Time, in microseconds, between pellet impact and received by the senssor	Roughly 300us for this design
Sensor N	Sensor mounted at the top most position (North)	
Sensor E	Sensor mounted at the right most position (East)	
Sensor S	Sensor mounted at the bottom most position (South)	
Sensor W	Sensor mounted at the left most position (West)	
V	Speed of sound in air,	343mm/ms @20C

3. **Introduction**

Electronic targets that capture a shot and display the results have been around for a number of years. There are no shortage of suppliers, but unfortunately, the cost is outside of the price range of most people.

3.1 Requirements

The requirements of this project are:

1. Uses a conventional lap top or PC for the display device
2. Signal processing is handled off-board on a low cost processor
3. Material cost to be less than $100
4. Open source to allow the project to evolve in time.

4. **Operation**

The fundamental problem to be solved is to detect the location of a pellet or bullet as it passes through a prescribed area. Once that is done, the presentation of the information is a relatively straight forward if not detailed matter.

There are a number of ways of doing this:

- High speed video recording
- Measurement through a light gate
- Acoustic location

Acoustic location has been used for many years and the technology is readily accessible to most developers. For this reason, acoustic location is used in this project.

4.1. Overview

The eTarget uses four acoustic sensors (microphones) arranged around the target as shown in Figure 1. The pellet moving over the sensors will emit a sound wave in a circular pattern that is picked up by the sensors. The sound radiates outwards uniformally at the speed of sound, So the time values of n, e, s, and w are proportional to the distance, and if the values are known, then we can compute the position of the pellet.

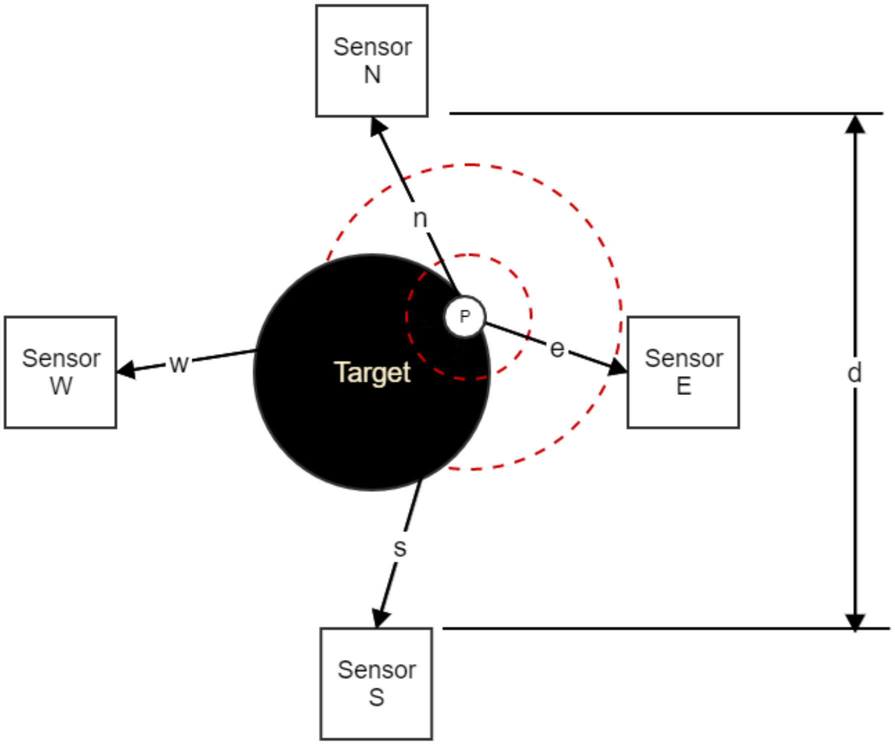

Figure 1: Acoustic Sensor Arrangement

4.2. Calculations

Using Figure 1 as a starting point, the relevent physical values are applicable.

- For an air pistol or rifle target, the distance between sensors is

  100mm

- Position resolution to identify a hit to within 0.1 point (ex 10.7 Vs 10.8) is

  Resolution = space between rings / 10

  = 2.5mm / 10

= 0.25mm or better

- Longest report time occurs if the pellet passes immediately in front of a sensor

  Time = distance between sensors / speed of sound

  = 100 mm / 343 mm/ms

  = 291 us

- Time resolution to locate the pellet is:

  Time resolution = distance resolution / speed of sound

  = 0.25mm / 323mm/m

  = 0.00072 ms

  = 0.7 us per 0.1 point

4.3. **Circuit**

Many implementations found on the internet use the analog conversion circuit of an Arduino or Raspberry Pi to measure the time between sensors. From the calculation above, this circuit is nowhere near fast enough or precise enough for the task at hand. The solution has to be to do the acquisition in hardware and the calculations in software.

4.3.1. Requirements

The time acquisition circuit for this project needs to meet the following requirements:

- Measure the sound from four sensors

- Measure the sound to better than 0.7 us. or better
  - For the purposes of our design, an 8MHz or 0.125us sample will be assumed

- Measure the sound for up to 291 us.
  - For the purposes of our design, 350us will be assumed

- Number of timer bits = log2(Maximum Time / Time Resolution).

  = log2(350 us / 0.125 us)

  = 12 bits or better

4.2.2 Block Diagram

The block diagram for the data acquisition circuit is shown in Figure 4.3

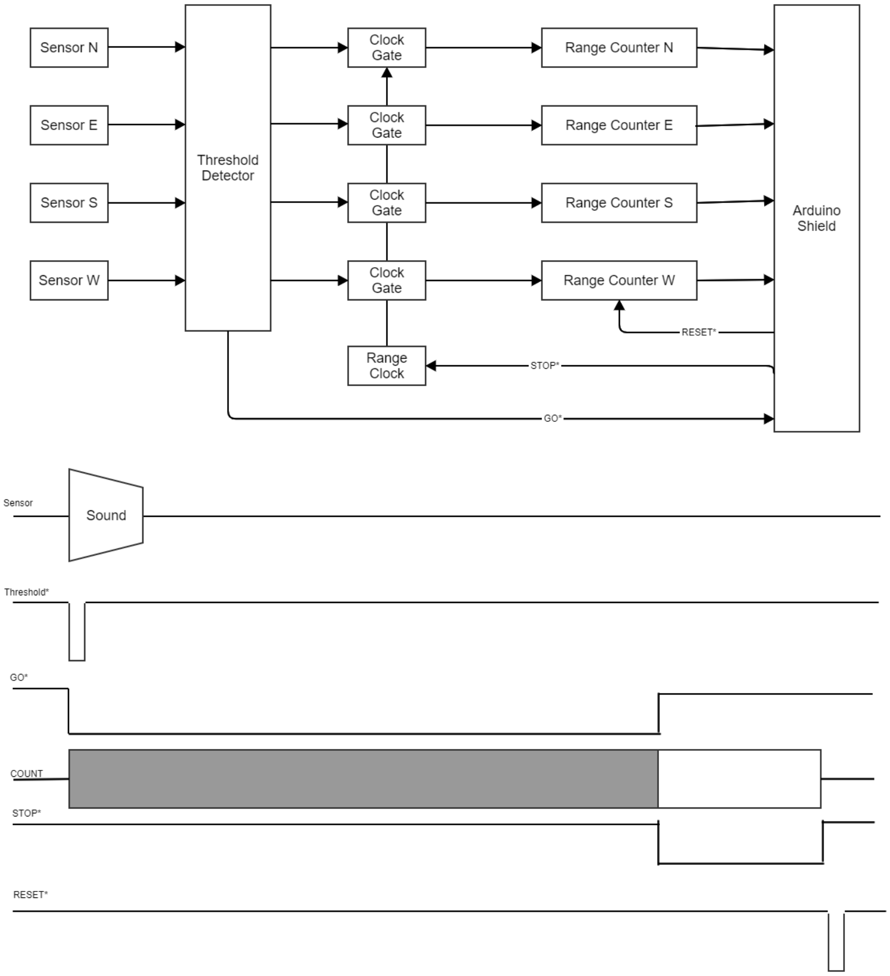

Figure 4.3: Acquisition Circuit Block Diagram

As the projectile passes over the sensor, the sound wave will generate a wave from the sensor that when it exceeds a threshold will generate a pulse which is the signal used to start GO* to begin driving the counters. A short time later, the Arduino generates STOP* to hold the count at their current values. The software then reads the values out, does it's calculations, and pulses RESET* to clear the counters and repeat the process.

5. Shot Location

Starting with Figure 1, the geometry of the shot detection is shown in Figures 5-1 and 5-2. The distances N, E, S, and W are the distances between the projectile and the sensors. The distance K is the shortest distance between the sides of the sensors.

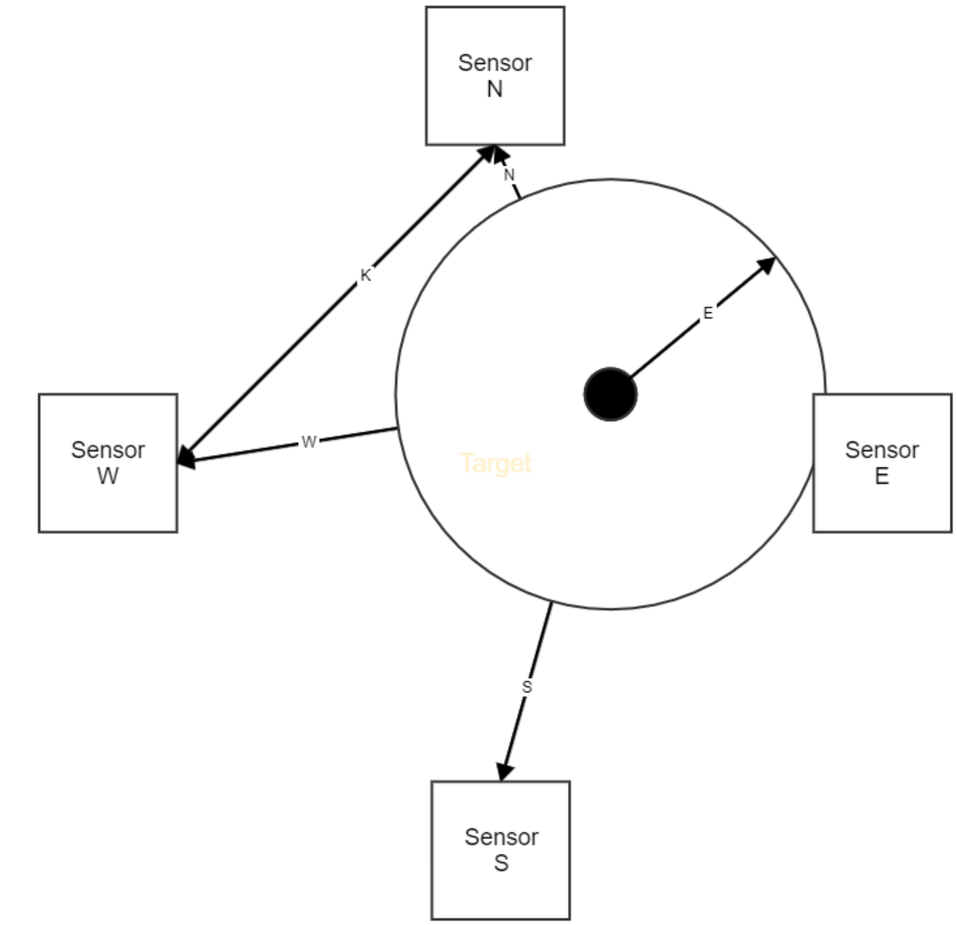

Figure 5-1: Shot Geometry, Sound Propagation

Sound travels out uni-formally from the projectile (distance e) until it reaches Sensor E and begins the Sensor e Timer. Subsequently, Sensors N, S, and see the sound wave and begin timing. When the timers are stopped, and the largest counter (Sensor E) is subtracted from the other counters, we are left with distances n, s, and w. Since the only unknown in this system is the distance e, our problem simplifies to finding the value of e that when added to n to w gives us the distances N to W and the location of the impact.

n = E -N

s = E - S

w = E - W

If we guess the value of e and insert it into the geometry of Figure 5-2 we get the four triangles shown in Figure 5-3. The base of each triangle is the known distance K, and the sides are the known counter values summed with the estimerate for E

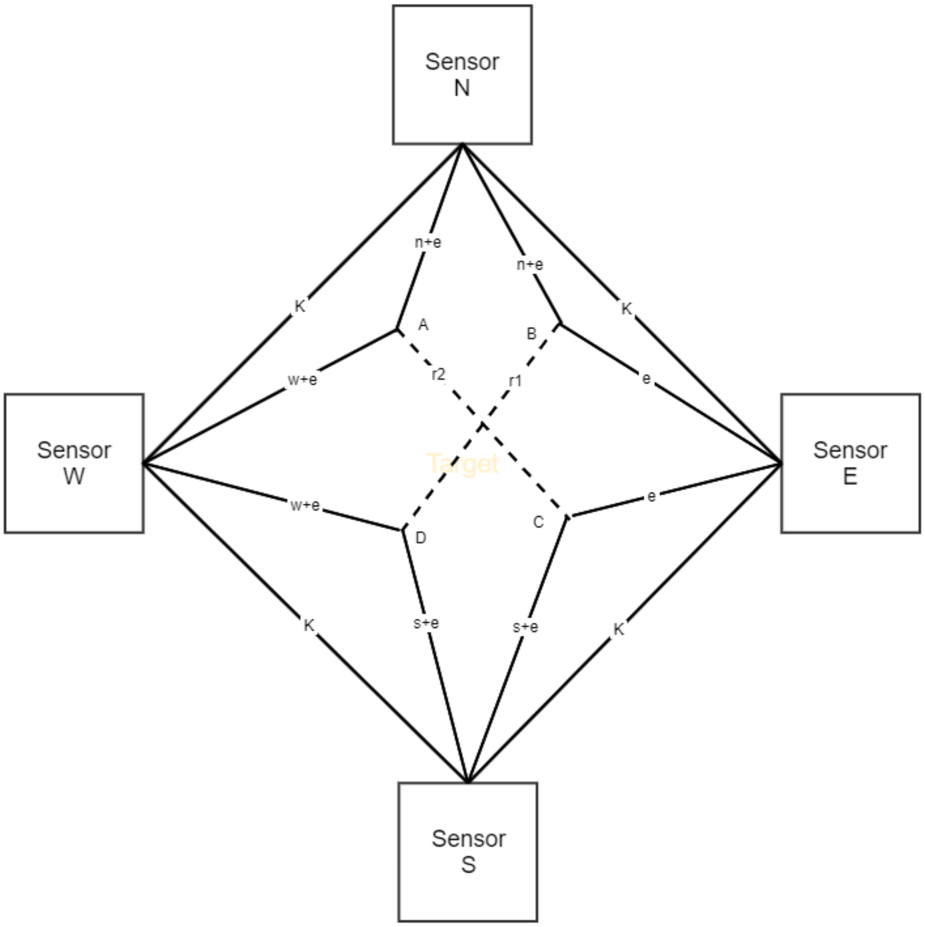

Figure 5-3: Shot Geometry, Triangle Intersection.

We can find the locations A, B, C, and D using trigonometry. For the purposes of illustration, triangle K-(s+e)-e is rotated 45 degrees and shown in Figure 5-4.

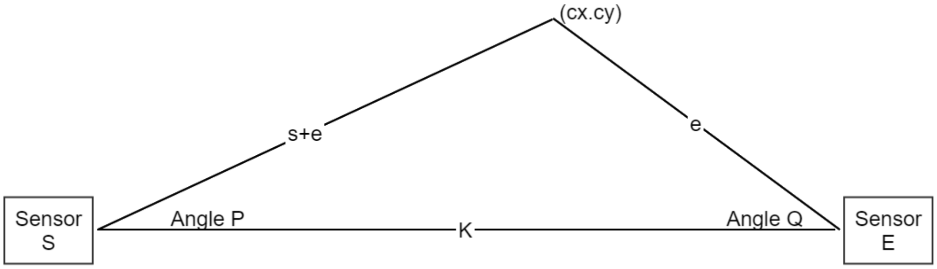

Figure 5-4: Intersection Triangle, Law of Cosines

From the Law of Cosines,

$$\mathbf{e^2 = (s+e)^2 + K^2 - 2(s+e)K \cos (P)}$$

Re arranging terms

$$P = \text{arcos}(e^2 - K^2 \ / (2(s+e)K))$$

By trigonometry

$$cx = (s+e) \cos(P)$$

$$cy = (s+e) \sin(P)$$

Rotating theta back into the original circle.

$$Cx = (s+e) \cos(P + 45')$$

$$Cy = (s+e) \sin(P + 45')$$

This process is repeated three more times for the pairs N-E N-W and S-W, yielding locations: (xn, yn) (xe, ye) (xs, ys) (xw, yw)

The distances r1 and r2 are then

$$r1 = \text{sqrt}((Ax - Cx)^2 + (Ay–Cy)^2)$$

$$r2 = \text{sqrt}((Bx - Dy)^2 + (By - Dy)^2)$$

So, the simplified algorithm for determining the projectile position is:

- Start timer(s) when the projectile crosses the sensor(s).
- Stop the timers one millisecond after the first projectile was detected
- Find the largest time
- Subtract the largest time from all of the timers
- Set the error estimate e to be 1 mm

```
e = 1 mm;
r1 = large number;
r2 = large number;
while ( (r1 + r2) > threshold )
{
    Compute locations (xn,yn) (xe, ye), (xs, ys), (xw, yw) based on the timers and the estimate (e)
    Compute distances r1 and r2
    error = r1 + r2
    e = e  + error/2
 }
```

- score = 10.9 - sqrt(xn**2 + yn**2) / .25