Aaron Xie and Abigail Liu
Los Altos Community Team 0399

Sensing the World: A Deep Dive into the Different Methods of Interacting with the Environment Around Us

I. Introduction

Sensors are a crucial part of Botball and are widely used in all areas of robotics. Parts like motors and servos enable the robot to move and turn. However, sensors are the eyes and ears of the robot; giving feedback on position, speed, and operating state to the controller. The robot can then react based on this feedback to get the robot to perform various tasks to precision. Without this feedback, the robot would be unable to react to its surroundings and complete its tasks. This paper will discuss all of the sensors allowed in Botball competitions, describe how the sensors work, illustrate some typical use cases, and help others in the Botball community find the right choice of sensor for their robot.

II. How Sensors Work

At their core, sensors convert one form of energy (the input) into another (the output). A touch sensor, for instance, transforms mechanical pressure into an electrical signal. These signals are then processed by microcontrollers or other systems to trigger actions or provide data.

Sensors can be classified as analog or digital. Analog sensors produce a continuous output that varies based on different inputs as well as from different ranges. The Tophat sensor, primarily for line-following, is an example; the sensor outputs a higher value when exposed to a darker surface and a lower value for lighter surfaces. Digital sensors in Botball, on the other hand, produce binary outputs (e.g., "on" or "off"). Touch sensors are a common example, outputting a signal only when pressure is detected.

III. Types Of Sensors & Applications in Botball

Touch Sensor (Digital Sensor, Binary Sensor):

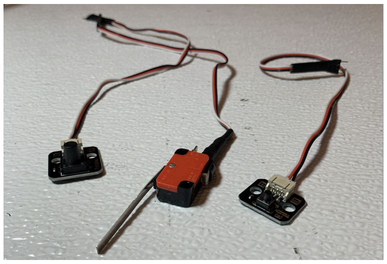


Figure 6 Touch sensors (Large Button, Lever, Small Button from left to right)

Touch sensors are digital sensors. They are also binary, meaning they have two states, 0 and 1, with the default state being 0. When triggered, the output value changes from 0 to 1, and it returns to its default state when the button or lever is released.

A common use case for the touch sensor is to mount it on the base of the robot. When the robot makes contact with another structure, the touch sensor is triggered, which stops the robot at the desired position.

There are three types of touch sensors in Botball. The lever touch sensor is primarily used for structures with larger or farther away surface areas. On the other hand, button touch sensors are used for more precise contact. Figure 7 shows a use case when the button touch sensor is needed to accurately locate the pole.

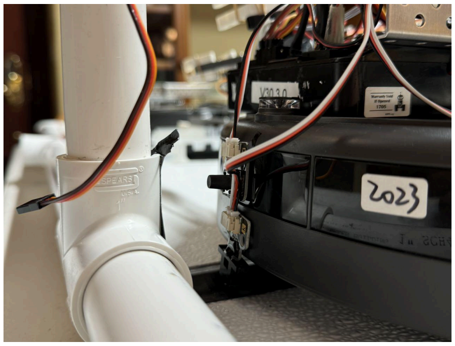


Figure 7 Using touch sensor to locate the pole

ET Sensor (Electromagnetic Tracking, Analog Sensor):

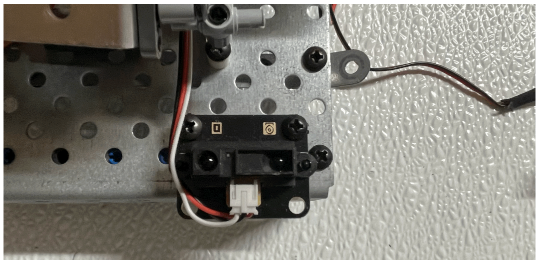


Figure 1 ET sensor

ET sensors can detect the distance between the sensor and an object. The reading reflects the distance between the sensor and the object, but it is not linear. One use of an ET sensor is to stop the robot a certain distance away from an object. This can be crucial when there is no other reference point, and knowing the robot's position is very important.

Another example is stopping when it senses an object that is in the way of the robot. This can prevent the robot from crashing into the object and ruining the run, allowing an arm or other structure to sweep it aside.

Table 1 Readings of ET sensor for different distances

	Reading 1	Reading 2	Reading 3
0.5 in.	2,464	2,215	2,216
1 in.	2,130	2,205	1,926
2 in.	2,904	2,907	2,908
3 in.	2,905	2,910	2,909
4 in.	2,730	2,731	2,772
5 in.	2,234	2,260	2,218
6 in.	1,903	1,925	1,928
7 in.	1,720	1,673	1,625
8 in.	1,485	1,508	1,484
9 in.	1,348	1,375	1,346
10 in.	1,232	1,253	1,256
11 in.	1,230	1,136	1,134
12 in.	1,047	1,100	1,033

Table 1 above demonstrates the exact readings of the ET sensors from different distances. To further illustrate, below is a graph comprising of the average values of above readings.

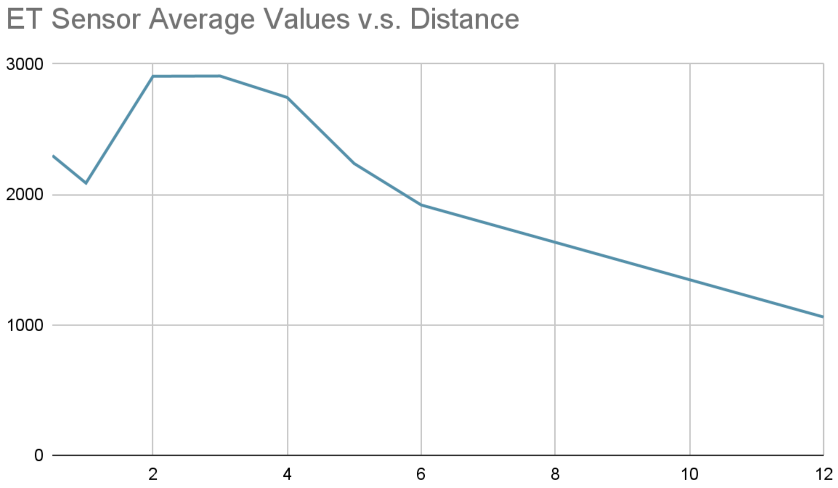


Figure 2 ET Sensor Average Values to Distance

As observed in Figure 2, the ET sensor value peaks out at about 2-3 inches range, and the value decreases as either the distance decreases or increases. This is because the ET sensor's focal point is ~4". When the distance is less than the focal point, the sensor value will decrease and we cannot tell whether it is too close or far away. Therefore, in order to correctly use the ET sensor, we need to make sure we use it for the correct range (4", 1m).

Camera (Analog Sensor):

Figure 3 Camera

Cameras can differentiate colors of objects in the camera's view and connects to the wombat via USB port. The camera has a display on the wombat, where the camera's view can be seen.

The camera can be configured and programmed to recognize a specific color through a widely-known computer vision library. When detecting objects, boxes appear over sections of different colors on the wombat display. The boxes represent the areas the robot sees the specific color. Quite often, many boxes of different sizes are returned, as the robot can see multiple areas of an object that correspond to the specified color, so it is recommended to do filtering based on the area of the box.

A frequent use case for the camera is for sorting poms, as the camera can differentiate between two colors to help the robot decide which pom to pick up.

Tophat Sensor (Analog Sensor):

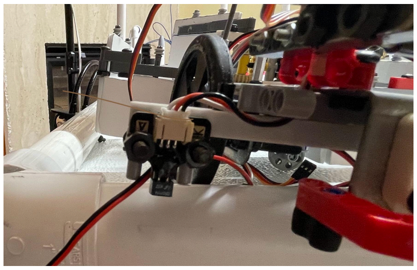


Figure 4 Large Tophat sensor

A tophat sensor is an analog sensor that emits infrared (IR) light at the surface it faces. It receives the amount of IR light reflected, calculates the value of IR light lost and retains it as its value[1]. Darker colors absorb more infrared while lighter colors absorb less, and therefore the smaller the amount of IR received results in the larger the value of the reading[2]. For example, the black tape on the Botball game board gives a value reading of about 3000, while the white particle board gives a value reading of around 190. Therefore, the Tophat sensor is mostly used to differentiate Black from White, the two major colors on the Botball game table.

We tested the Tophat sensor in various environments and have the following findings:

(1) In terms of value difference between Black and White, the tophat sensor is not affected by environmental lighting nor surface temperature. We tested sensor reading on 1) warm vs cold surfaces; 2) artificially lighted room vs unlighted room. We found that the difference between the white value and black value are very similar in all cases.

(2) Most objects (with varying colors other than black) are equivalently absorptive to the infrared light of a Tophat, as depicted in Table 2. Note that the extra small value of "53" results from two tophat sensors shooting IR at each other. Because both sensors receive around the original amount of IR, the amount absorbed/lost is very little (due to the glass), hence the reading is very low. This confirms that the reading measures the amount of IR absorbed. Based on the specification[3], the wavelength of the tophat sensor is 940 nm as shown by Figure 5.

Table 2 Tophat readings for various colors

Object/Color Being Detected	Value
Red	199
Orange	203
Green	211
Dark Blue	193
Light Blue	183
Different Tophat	53
Black	3103

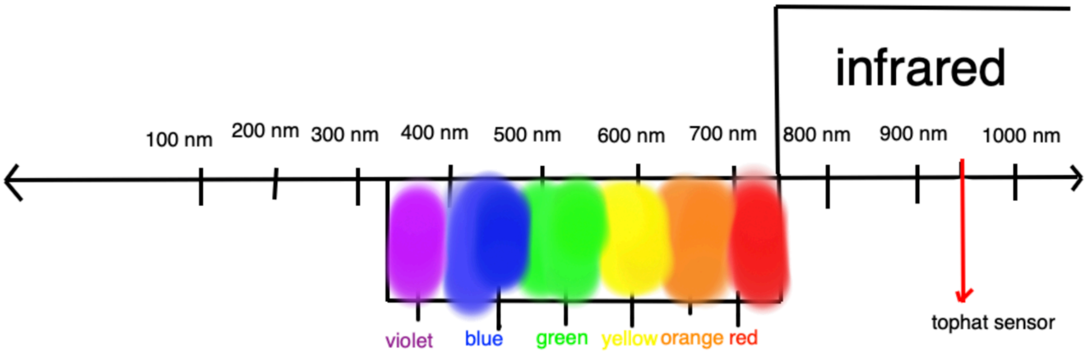


Figure 5 How Tophat works in electromagnetic spectrum

A common use case for the tophat sensor is for line follow. By placing the sensor on the intersection between the black tape and white board, the robot is able to follow the line by shifting to the right or left. If the reading is greater (meaning more "black"), the robot would shift towards the white area and vice versa. A larger tophat sensor is optimal for line follow due to its size and ability to detect the edge of the black line, while a small tophat sensor is well suited for locating the edge of a line for a "drive until line" code.

Because the IR light strength may vary between different locations, calibration of the tophat sensor is recommended to avoid errors.

Light Sensor (Analog Sensor):

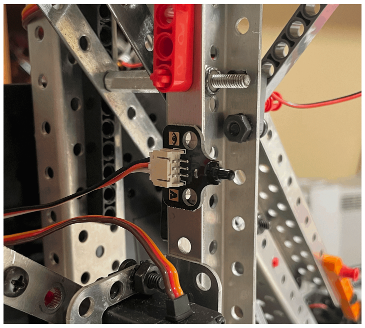


Figure 8 Light sensor

Light sensors detect the strength of a light through a large range of values. One common use case for a light sensor is when a light is shone to signal the start of the competition. A light sensor can detect this change and send a signal to the wombat to start the robot.

As an analog sensor, light sensors need to be calibrated before use. From our team's practice, the improved calibration requires the operator to shine a light into the sensor and click a button to read values. For this to work, the light needs to be turned on to register an input of what a light-on value looks like into the sensor. Following this, the light needs to be turned off to save a new value. This represents an input of what a light-off value looks like. A threshold value is created between the light-on and light-off values. When the light sensor detects a light value that surpasses the threshold, the sensor knows the light is on and notifies the robot.

$Th = 0.7(X - Y) + Y$ where,

Th is the threshold value that decides whether the robot starts

X is the light-on value

Y is the light-off value

Calibration is required as certain environments are brighter than others. This yields different light-on and light-off values. Therefore, this calibration equation will be used to place the threshold value where it needs to be to adjust to ambient light.

Linear Slide Potentiometer (Analog Sensor):

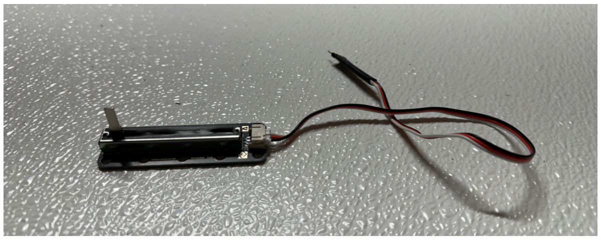
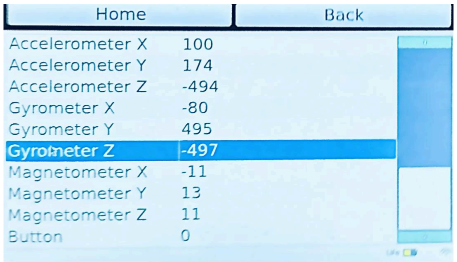


Figure 9 Linear Slide Potentiometer

The linear slide is an analog sensor that detects the position of the slider on the slide. The sensor ranges from 0 to 4086, allowing for a precise measurement due to its small size.

A linear slide is commonly used as a horizontal elevator. By attaching a motor onto the slider, the robot is able to track the position of the structure more accurately than just the motor itself. Another possible use case is splitting the slide into multiple ranges and therefore converting the continuous analog signal into a discrete signal. When the slider is moved, different values can be coded for different actions.

Gyroscope (Analog Sensor):



```
Accelerometer X    100
Accelerometer Y    174
Accelerometer Z    -494
Gyrometer X        -80
Gyrometer Y        495
Gyrometer Z        -497
Magnetometer X     -11
Magnetometer Y     13
Magnetometer Z     11
Button             0
```

Figure 10 Gyro sensor (embedded in Wombat)

The gyroscope is one of the most useful sensors in Botball movements, as it ensures the robot drives straight or turns precisely. Gyroscope tracks the instantaneous angular velocity of the robot. The integral of angular velocity becomes the angle change. For example:

$$angle\_change(t) = \int gyro\_ticks\, dt \text{ , where } t \text{ is } time$$

The wombat offers three axes that are mutually perpendicular to each other. These axes are known as x, y, and z. However, typically only one axis is used. The axis chosen depends on the orientation of the wombat.

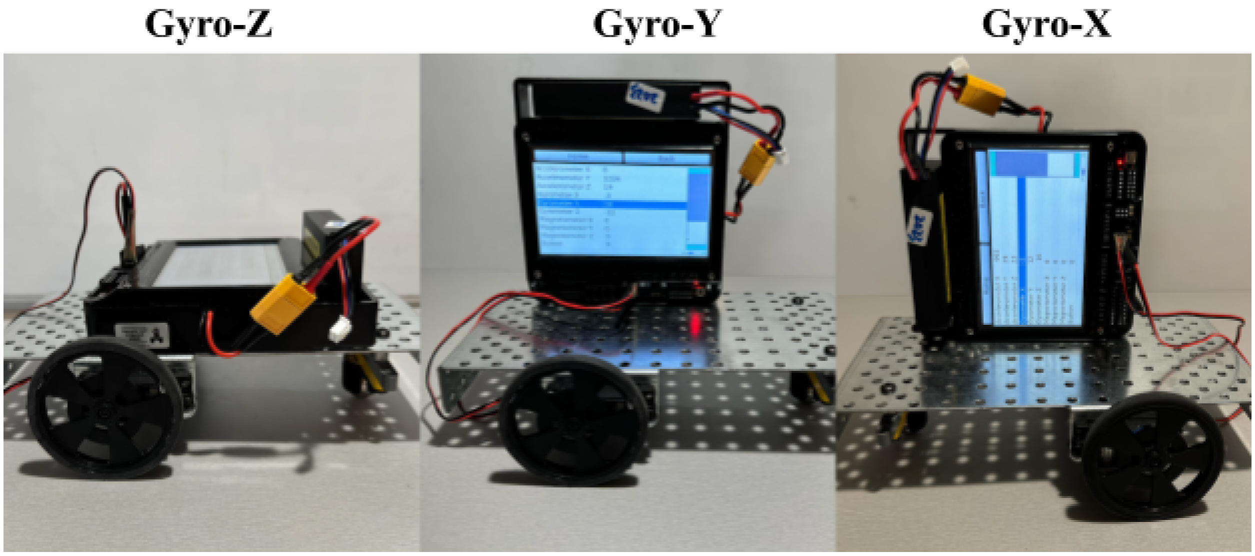
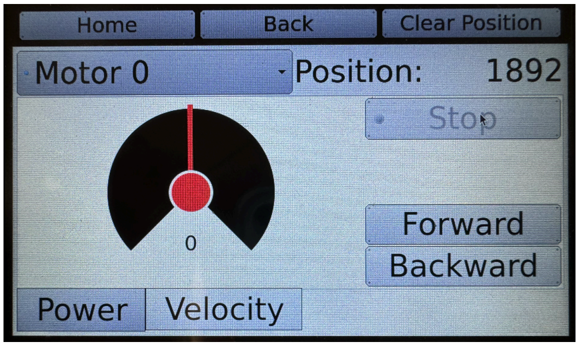


Figure 11 Using different Gyro axes depending on Wombat mounting

Figure 11 illustrates the different axes. In this image, the z-axis aligns with the movement of the robot wheels. Therefore, it was chosen in Figure 10. We can use the gyro reading integrated over time to determine the change in angel from the desired direction[4]. This can be used to guide the robot's wheel speeds to bring the robot back on track, a useful feature when "line-follow" is not available.

Motor position counter (Analog Sensor):



Motor 0
Position: 1892

Figure 12 Motor position counter (embedded in Wombat)

The motor position counter (MPC) keeps track of how far the robot has moved. This distance is measured in "ticks". Similar to how a clock has 60 ticks per hour, a motor has ~1800 ticks per rotation (360 degrees). The number of ticks per rotation may vary between motors, and it's not affected by the size of the robot's wheels – just like the length of a clock's hands does not change the number of ticks per hour.

The most common usage of the motor position counter is to drive the robot for a specific distance. Here the distance is defined as a motor position, which can be calculated in the following formula:

$$goal\_position = 1800 * distance / (PI * wheel\_diameter)$$

```
void line_follow(int port, int speed, int goal_position){
   cmpc(MOT_LEFT);
   cmpc(MOT_RIGHT);
   while(abs(gmpc(MOT_LEFT)) < goal_position) {
      double frac = (analog(port) - WHITE_VAL) / (BLACK_VAL - WHITE_VAL);
      int l_speed = speed - (frac - 0.5) * speed;
      int r_speed = speed + (frac - 0.5) * speed;
      drive(l_speed, r_speed);
   }
}
```

Figure 13 Code example of line-follow until distance using MPC

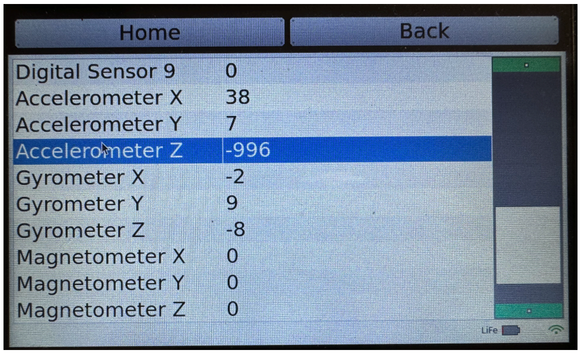
Note that the code uses *abs(gmpc(MOT_LEFT))*. This is because the motor position counter can have negative values when the motor moves backwards.

Besides driving the robot, sometimes we need to use motors instead of servos to control robot arms due to part limitations. Different from servos, motors can revolve infinitely, and hence they do not have a predefined position. When we use a motor to move an arm, we cannot do a simple *set_servo_position()* to complete the arm movement. Instead, we need to clear the motor position counter (remember it revolves infinitely), and move the motor until it has the same position counter as the destination. The code example is shown in Figure 14.

```
void motor_to_position(int port, int speed, int goal_pos){
   cmpc(port);
   while(abs(abs(gmpc(port)) - abs(goal_pos)) > 5){
      mav(port, speed);
   }
}
```

Figure 14 Code example of turning motor to control an arm

Accelerometer (Analog Sensor):



Digital Sensor 9	0
Accelerometer X	38
Accelerometer Y	7
Accelerometer Z	-996
Gyrometer X	-2
Gyrometer Y	9
Gyrometer Z	-8
Magnetometer X	0
Magnetometer Y	0
Magnetometer Z	0

Figure 15 Accelerometer (embedded in Wombat)

The accelerometer sensor tracks the acceleration of the robot. Similar to gyroscope sensors, an accelerometer sensor has three axes in wombat: x, y, and z. The physics behind the usage of accelerometer is the same as gyroscope sensors: we integrate continuous reading of the sensor into different quantities. For accelerometer sensors, the integral of acceleration becomes velocity change.

$$velocity\_change(t) = \int accelerometer\_ticks\; dt \quad \text{, where } t \text{ is } time$$

However, velocity itself is not very useful in Botball. We usually have to integrate velocity further into position or angle. The two levels of integral could cause loss of accuracy. Small errors in acceleration may be accumulated into larger errors in the velocity and position estimates. Therefore, accelerometers are not commonly used in Botball. When we need to accurately track the angular movement of the robot, we use gyroscope sensors.

IV. Conclusion

All the aforementioned sensors are vital to both Botball and everyday technology. Each type of sensor has unique benefits and roles to play in completing tasks on the game table. From guiding our robots to their task areas to scanning game board pieces, sensors were crucial in aiding our robots to complete their tasks. In this paper, we researched how sensors work, analyzed the performance between similar sensors, and suggested their common use cases and limitations. Having a deep understanding of sensors will aid teams in scoring more points, and we hope this research can benefit the Botball community as a whole.

References:

[1] Devasia, Anish. “Infrared Sensor: What Is It & How Does It Work.” *Safe and Sound Security*, 27 Sept. 2023, getsafeandsound.com/blog/infrared-sensor/#:~:text=Active%20Infrared%20Sensors&text=They%20work%20by%20transmitting%20infrared,the%20object’s%20distance%20and%20presence.

[2] “Line Sensors and How to Use Them.” *FutureLearn*, 25 Oct. 2022, www.futurelearn.com/info/courses/robotics-with-raspberry-pi/0/steps/75899#:~:text=Infrared%20(IR)%20detection,example%20TCRT5000%20line%20sensor%20below.

[3] *Reflective Object Sensor QRD1113, QRD1114*, www.onsemi.com/pdf/datasheet/qrd1114-d.pdf.

[4] Brain, Marshall, and Desiree Bowie. “How the Gyroscope Works.” *HowStuffWorks Science*, HowStuffWorks, 7 Sept. 2023, science.howstuffworks.com/gyroscope.htm#:~:text=This%20mysterious%20effect%20is%20precession,axis%20of%20your%20applied%20force.