Referencing your robot on the gameboard

Improvements and advice

Michael Eder
HTL SAALFELDEN
Higher Technical College for mechatronical engineering
Saalfelden am Steinernen Meer, 5760 Austria
michael.eder@htl-saalfelden.at

Florian Russegger
HTL SAALFELDEN
Higher Technical College for mechatronical engineering
Saalfelden am Steinernen Meer, 5760 Austria
florianrussegger98@googlemail.com

***Abstract*—Not knowing where your robot is positioned is a major problem. This document includes ways to reference your robot on the gameboard. We have considered every possible solution to this problem. Ways to reference the movements of the effectors are also covered. Furthermore, suggestions and improvements will be listed.**

Keywords—referencing; sensors; debouncing; game board; game objects;

I. Introduction

As referencing is a key part of the botball tournament, we have taken a closer look at the ways to reference efficiently on the gameboard. What is referencing you might ask? Referencing means that your robot always checks its position. E.g. measures the distance to a fixed object, drives along a black tape or uses limit switches. But also, the robot should be able to operate its effectors reliably by using sensors. The robot should be able to perform the programmed task nearly every time. Therefore, should be able to adapt to unknown errors and uncertainties. We have looked at the various sensors that are available in the game kit and how these could be applied in the tournament. The way the gameboard is currently set up has also been considered. Furthermore, the paper should provide an instruction manual for future botball teams on how to get the most out of their parts.

II. Hardware Soloutions

One part of referencing on the game board will be to recognize various game objects (hayballs, bins, etc.) and the gameboard (game border, tape, ramp) itself.

A. Referencing from the border tubes

One of the best methods of referencing your robot is to do it with the tubes which frame the game board. As the tubes are fixed to the game board it’s possible to use every sensor available. By measuring the distance with the ET Sensor, you are able to use them as limit switches for your desired position on the gameboard. As explained later in more detail, the accuracy of the sensor heavily depends on the light conditions in the room. Another type of sensor which can be used would be a mechanical switch which refers mostly to the Create robot as it has them already build in. These sensors, if there is no mechanical issue, give very reliable results with the disadvantage that you have to have a border tube nearby. Another easy way to reference your robot would be to drive against the border until your tires slip on the gameboard. If you don’t have an oddly shaped robot it will align itself perpendicular to the border every time. This creates a new, partly defined position which gives you perfect opportunities for your next driving actions.

B. Referencing from the black tape

The black tapes are, if there are any, a very accurate way to reference your robot. The main problem we had to face was that there were too few lines on the table to only rely on them because you are only able to detect them if you are standing right above them. This concludes that they are only usable for limit switches or to drive along them if you are lucky and your desired path matches with the tape. Another problem is sun or bright room light which reflects on the white gameboard. A simple and obvious solution would be to install the sensors in the robots own shadow or create an artificial one.

1) Drive along the black tape

If you are willing to drive along the black tape you have the option to use one or two tophat sensors. In the case of 1 sensor its best to keep on one edge of the tape. This brings up the problem that the side of the tape the robot chooses to drive is really hard to detect which may end up in unwanted results.

The method we use and recommend is that you use two tophat sensors which are mounted parallel, in a distance to each other that if the robot is placed centred on the tape both sensors are black. Now you can detect the side the robot has left the tape and correct the direction in which the robot is heading. After loads of testing we prefer this method because on the one hand its way cleaner and easier to program and on the other hand it is more robust to inaccuracy’s in the pre-positioning of the robot.

C. *Referencing from game objects*

Game related objects can really help you with referencing your robot. One possible way of detecting the objects is to use your ET sensor. Mechanical sensors are only suited in special condition especially if the objects are unable to push away. In the case of the ET sensor, reflections of sun- or bright room light caused by the transparent bowls or the gamebord itself, make it hard to get correct readings out of it.
To prevent disturbance of the sensor readings we used several methods to shield our sensors.

- Paper funnel: roll up strip of paper into a tube and fix it to the sensor with a rubber bands or tape.
- To prevent any hassle with the paper you can use the LEGO® tire perfectly for shielding the ET sensor.

D. *Internal referencing*

The main goal of internal referencing is to supply the effectors of your robot such as grippers, linear actuators etc. with limit switches, object detection or analog sensors. This is especially important if the robot relies on motors. When the battery gets low the motors won't turn on full speed anymore or if you jump over some gear teeth because of high torque.
Analog sensors such as the slide resistor are tremendously helpful by determining component positions not only in their limits but on the whole way of action. This is a huge improvement, because a two-way communication is established. This means there is one component which transmits the force and an unloaded component which sends back the position.

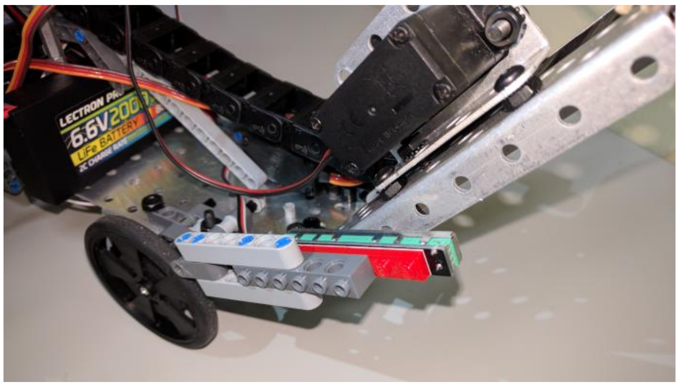
Fig. 1 linear slide used to determine the position of the arm

The small buttons are almost impossible to install because there is no place to attach a LEGO® part or a screw. The new 2017 sensors are a huge improvement to the old ones but unfortunately, we haven't seen any changes in the button compartment yet.
For building a botball robot the target would be to leave as little undefined positions as possible.

III. Software Soulutions

On the other hand, the various sensors would be useless without good programming and proper ways to measure them. Also, there are ways to reference your robot software wise, without the need of external game objects. In the following sub-items, numerous methods will be listed and discussed.

A. *Referencing via motor ticks*

The included botball motors can drive with a certain velocity or to the desired motor ticks. The main problem with driving only at a certain velocity is quite clear. The emptier the battery gets the weaker will the motors turn. Therefore, won't be able to go as far if a timer is set. This is where the ticks come in handy. The motors won't be dependent on the battery load. The motor will turn quite exactly to the desired ticks. Still a major problem remains. If the robot hits an object, the wheels will turn and the ticks will be reached before the required position on the gameboard.

Another problem occurred. The motors will never be able to turn at the same speed because of their manufacturing precision. Resulting in a far from straight path. This is where the motor ticks come to shine again. By querying the motor ticks on one motor we can correct the other motor.

Example: The right motor finishes its ticks first. Then it will reduce its velocity, therefore brake a little. Just enough for the left motor to finish its ticks. Then they will repeat this the desired amounts of times. The outcome is an extremely straight path.

This method can also be used for precise turning in the same fashion. Resulting in a precise turn with braking at the end.

We have marked a straight tape. The robot was always placed at the same spot and in line with the tape. Then we wrote a program to drive 1 yard straight ahead. Then we were measuring the offset from the tape. We only have positive offsets. A positive offset represents an offset to the right. The average of our measurements is at 2 cm offset (Fig. 2). Which is definitely good enough for our needs. We've also tried the same experiment with only driving by time (Fig. 3). The average offset was significantly higher at 3,5 cm. Although the standard deviation is at 1 cm both times, our method of driving via motor ticks is more effective.

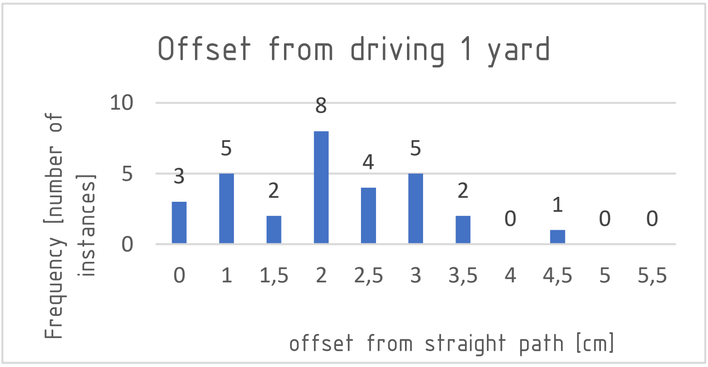

Fig. 2 Offset from driving 1 yard via motor ticks

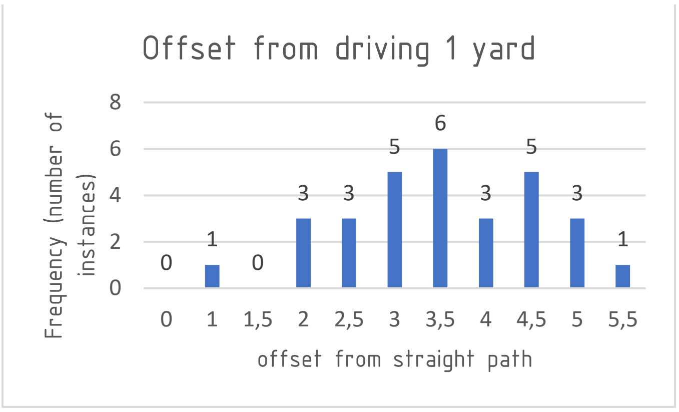

Fig. 3 Offset from driving 1 yard via time

B. *Referencing by time*

One simple and easy method to get to the desired location is to just let the motor turn for a certain amount of time. Therefore, leaving the traditional referencing out. But it is to mention that this easy way of moving on the gameboard is quite risky. If there is a slight bump the wheels will turn nonetheless. This is the result of the robot not being able to adjust its path, due to the lack of referencing. If the next step in the program relies on a sensor value, the robot might be too far off to execute it properly. Thus, it will be extremely hard to know the location of the robot. To get around these issues it is important to have a completely charged battery, because we've noticed that the motor velocity is battery-dependent. So, a slower velocity and the same time result in a shorter travelled distance as in equation (Equation 1).

$$s_{charged} = v_{charged} * t > s_{discharged} = v_{discharged} * t$$
$$v_{charged} > v_{discharged}$$

Equation 1 comparing charged and discharged

C. *Software to compliment a reliable robot*

Using incoming sensor values without "debouncing" is sheer useless. If a sensor stands still its value still jumps ±50. Debouncing is a term out of digital electronics. When a button is pressed, it will bounce a few times before it reaches its final state (high/low). As seen in figure 4. The same problem occurs with the optical sensors. To get rid of this problem the sensor must reach a threshold more than once. This results in a reliable sensor. The following code example (Code 1) shows how a sensor could be debounced.

```
/*------------------------------------------------

debounce function by Johannes 2/25/17

-------------------------------------------------

Function reads a distance sensor value and debounces that
value. Because values of the distance sensors are jumping
alot it is key to debounce the incoming signal. Therefore, if
the sensor is bigger than int distance a counter is increased
by 1. If the sensor value drops under int distance the counter
is decreased by 1.

------------------------------------------------*/

void debounce(int distance, int sensor, int speed)

{

    int i=0 ;                     //counter

    int drive_l=1,drive_r=0;

    while (i<5)        //repeat till counter is bigger than 5

  {

    motor(drive_l,speed);         //motor_left turns

    motor(drive_r,speed);         //motor_right turns

    msleep(20);

    if(distance>analog(sensor)) //checking if sensor value
                                  is smaller than distance

    {

    if(0<i)            //if counter is bigger than 0

            {          //(to avoid negative counter)

            i--;       //counter gets decreased by 1

            }

    }

    if(distance<analog(sensor)) //checking if sensor value
                                  is bigger than distance

    {

    i++;               //increase counter by 1

    }

  }

  ao();                //turn all motors off

  msleep(50);

}
```

Code 1 Example program for debouncing a sensor

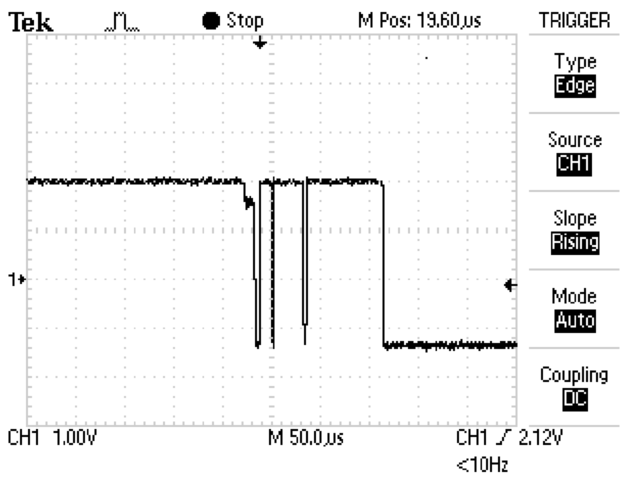

Fig. 4 switch being pressed; bouncing a few times before reaching the low state

Another important part is to never drive the motors at full speed. And instead of turning them off, slow them down. Otherwise the motors will turn even after they had been turned off. This may lead to unprecise turns and too long paths.

One more issue we've come across has to do with the servos and them not being able to turn at the needed speed. They are either off or at full speed. So, a simple while loop will do the job to move the servos incrementally with a short delay between increments.

D. Internal sensors

Internal sensors such as the gyro sensors or the acceleration sensor can also be used to reference your robot. With the gyro sensor you are able to detect the start as well as the end of the ramp. One thing to keep in mind is that the gyro sensor and the accelerometer only detect a deflection. Thus, the program unfortunately must utilize the deflection and not an absolute value.

IV. SUMARRY

As a conclusion, we think it is appropriate to say that the whole development was more or likely trial and error! So, we suggest upcoming botball teams to thoroughly test each and every sensor and decide which ones to use. This paper should provide a good view of some possible ways to reference your robot. It should be an advice rather than an instruction.

V. REFERENCES

This paper is only based on our own experiences throughout the building process of our robots. Every aspect presented in this paper has been encountered in the process of the botball tournament. Experiments have been done by ourselves and the collected data has been evaluated by team members only. No external help was used.