Turret Systems 101

Project Report

SW507E16

Aalborg Universitet
Institut for Datalogi

Copyright © Aalborg University 2014

AALBORG UNIVERSITY
STUDENT REPORT

Computer Science
Aalborg Universitet
http://www.aau.dk

Title:
Turret Systems 101

Theme:
Embeded Systems

Project Period:
Fall 2016

Project Group:
SW507E16

Participants:
Christoffer Mouritzen
Jonathan Magnussen
Jonas Ibrahim
Emil Hedeholm Sørensen
Simon Højberg
Michael Sebastian Søby

Supervisor:
Brian Nielsen

Page Numbers: 17

Date of Completion:
October 5, 2016

Abstract:

To Do

The content of this report is freely available, but publication (with reference) may only be pursued due to agreement with the author.

Preface

Aalborg University, October 5, 2016

Christoffer Mouritzen
<cmouri14@student.aau.dk>

Simon Højberg
<shajbe14@student.aau.dk>

Jonas Ibrahim
<jibrah14@student.aau.dk>

Emil Hedeholm Sørensen
<ehsa14@student.aau.dk>

Jonathan Magnussen
<jnma14@student.aau.dk>

Michael Sebastian Søby
<mss14@student.aau.dk>

Contents

Chapter 1

Introduction

Computers are an integral part of modern society, they are used in everything from assemblyline factories to kitchen appliances. More and more they are being used in daily life, without even being noticed. A computer now covers much more than just a desktop, laptop, tablet or a server. There is an entire class of computers surrounding us in much greater numbers, although their presence is a lot more incognito. These computers are built into larger electronic devices. This class of computer systems is also known as Embedded Systems.

Initial Problem

The theme of this semester is Embedded systems. This means that the focus of the project will be working with a set of constraints that must be balanced. Furthermore the project should include techniques from the machine intelligence course and the real-time systems course.

One of the project proposals was a challenge by René to build something that is able to hit a moving target. Based on our desire to learn from the experience as well as the reason “shooting stuff is fun” it was decided to work with this project. This produces the question:

Is it possible to construct an embedded system that is able to fire and hit a target in motion without user-input?

Chapter 2

Problem Domain

Fluff and red thread (Go indepth with the phases)

Ideal Scenario

Based on the initial problem it is now possible to describe how the turret is supposed to operate. An ideal scenario is constructed in order to better visualize it, and to serve as a basis for the requirements engineering. The ideal scenario involves the two actors “Turret” and “Target” and describes how they interact with each other.

Ideal Scenario:
The turret is set down on an even floor and turned on. It then turns searching for a target by turning left and right scanning an area for possible targets.
If the turret detects a target, it should be able to predict the targets movement, in order to anticipate the targets position at any given time.
The target is moving on an even plane at a constant speed driving in a straight line(perpendicular?). The target moves at a speed such that the turret will miss if it fires when the target is directly in front of the turret.
The turret moves into a firing position based on the predicted position and then fires at the appropriate time, in order to hit the target at the predicted position.
If the turret stops sensing targets, it should start scanning the area again.

Phases

Based on the ideal scenario three distrinct phases are identified, “Scanning”, “Tracking” and “Shooting”. The “Scanning” phase was where the turret searched for a target by turning. The “Tracking” phase where the turret collects data and calculates the trajectory for the target. The “Shooting” phase where the turret moves to a firing position and shoots at the correct time. These phases will now be examined in order to identify the problems in their respective domain.

Scanning

The “Scanning” phase which is used when the turret is idle, monitors an area. This presents two different situations. The first situation is when no target is detected and it continuosly scans for a target. The second situation is when a target is detected and it switches to the “Tracking” phase.

This leads to the following problems:

- How can an area for the turret to monitor be defined?
- How does the turret identify a target?

Tracking

The “Tracking” phase which occurs when a target is found, is used to follow a target and gather data about its movements. This information is then used to predict where the target will be at any given time as well as to calculate a shooting position.

This leads to the following problems:

- How can the turret predict a targets movement?
- How does the turret track the target?

Shooting

The “Shooting” phase occurs when it is determined where the target will be, when it will be there and how the turret will hit it. The turret then moves into the a shooting position and proceed to shoot at the correct.

This lead to the following problems:

- How does the turret move into a shooting position?
- How does the turret shoot?
- How does the turret hit a target a varying distances?
- What deadlines does the system need to have in order to hit a target?

Summary

In conclusion, an ideal scenario was constructed based on the initial problem. Certain hard requirements can be determined already based on the semester topic

1. The system must be embedded
2. The system must use techniques from either the MI or the RTS course, or both

In addition several problems were identified in the problemdomain:

- How will the cannon detect the target?
- What information is needed to properly calculate and predict the trajectory of a target
- What information is needed in order to calculate a firing-position and firing-time of the cannon in order to reliably hit the target
- How does the turret move into firing position?
- How does the turret shoot the target
- How does the turret hit targets at varying distances

These questions will serve as the foundation for the analysis, and help to refine the initial problem into a problemstatement. In the next chapter the theory behind embedded systems will be explained in order to properly understand the constraints imposed by such a system.

Part I

Analysis

Chapter 3

System Description

This chapter will be used to look at embedded systems as well as their limits, real-time systems concerning how scheduling and deadlines work and machine intelligence. - Add more text

Embedded Systems

An embedded system is a device with limited resources both in terms of computer power and how much memory it has. They are used everywhere to do small task and jobs. It is difficult to give an exact definition of what an embedded system is, but a crude definition[1, ch.1.1] is:

An embedded system is nearly any computing system other than a desktop, laptop, or mainframe computer.

This means that embedded systems cover alot of different types of computers and that requirements can differ alot between systems. They do share some similarities, these characteristics are:

1. **Single-functioned**: Most embedded systems only have a single function which is executed repeatedly.
2. **Tightly constrained**: All systems have some limits, but embedded systems due to their nature are much more tightly constrained when it comes to cost, size, performance and power.
3. **Reactive and realtime**: A lot of embedded systems also has to react and respond to certain input. If this input has a deadline, it is in fact a real-time system.

The constaints embedded systems have in terms of cost, size, performance and power dictates what they are used for, see Figure 3.1. The different features are linked to each other, if one is increased another should be decreased.

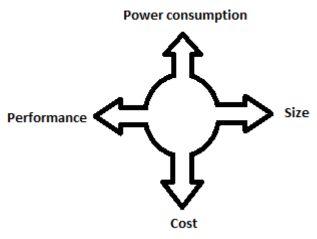

Figure 3.1: How the constaint can depend on each other

write about microcontroller?

Chapter 4

Platform Choice

In this chapter the two choices for hardware platforms will be presented. An analysis of the differences in the platforms will be conducted and in conclusion a platform will be selected for the system. The two hardware platforms suggested are

- Arduino
- Lego NXT Mindstorms

Lego NXT Mindstorms

The lego NXT Mindstorms is the successor to the first robotics-platform released by Lego the Robotics Invention System. Although the NXT line also contains 2 more recent platforms the NXT 2.0 and the EV3, in this project only the NXT 1.0 will be considered since this is the platform being offered by the university.The Lego

NXT Mindstorms kit consists of the NXT brick and a set of compatible sensors and actuators.The NXT brick consists of

- ARM 32-bit processor running at 48MHz with 256KB Flash storage and 64KB RAM
- 8-bit AVR running at 8MHz with 4KB Flash and 512B RAM
- 100x60 pixel LCD display
- Four buttons
- Speaker that plays sound files with a sampling rate of 8kHz
- 4 input ports and 3 output ports

The Lego NXT comes with the NXT-G programming environment which is a visual programming language for use with the NXT platform. Although there exists a variety of unofficial programming languages for the NXT, so in practice you can write programs in pretty much all the big programming languages. Finally some pro's and con's associated with the NXT as a platform:

- The external peripherals come programmed and assembled - all the user has to do is handle the data
-

Arduino

The arduino was born as a tool for fast prototyping, aimed at people lacking a background in electronics and programming. Over the years the arduino has evolved from offering simple 8-bit boards to offering a variety of different boards suiting different needs such as memory available, processing power, number of ports on the platform, etc. The arduino comes with its own programming language and an IDE. A large number of modules are available for the arduino allowing the construction of diverse and complex systems.

- Due to the popularity of the arduino platform a large variety of extern peripherals are available for the platform
- Due to the popularity of the platform, there is also a large amount of prewritten libraries for use with the modules
- If the library doesn't exist or contain the necesary functionality, it will have to be written, thus time will have to be invested in order to make it work

Sensor Testing

To find out which sensors are the best to use, this chapter will be used to test the different sensors. The testing of the sensors will also be used to test how precise they are.

First the testing testing methodology will be explained, then this methodology will be used to devise tests for the different sensors.

Testing Methodology

The testing will be divided into to two parts, one part is testing the two distance sensors and the second part is testing the camera.

Ultrasonic Sensor

The test of the ultrasonic sensor was divided into 4 different experiments, the difference between the experiments are the size of the object being tracked and whether the sensor is shielded. The different object sizes was tested with a shielded sensor and a non shielded sensor.

This sensor has a range of 0 to 255 cm and will not be too limiting in terms of how far the object can be from the canon.

Infrared

The test of the infrared sensor was not able to be conducted since there were some issues with the different sensors. The first sensor did not work at all, since it was an infrared sensor a camera was used to see whether the infrared LED lid up. Since it did not light we concluded that the sensor was broken.

When trying to test the other sensor, first it did not give a reading so the same test was conducted using a camera to see if the sensor was broken. It was concluded that the sensor was not broken. It was then discovered that it was a problem with the software, where the address for the reading the sensor on the I2C port was changed. This meant that it was not possible to get a proper reading. The readings were either 0 or a reading above 60.000 depending on which port was used.

Another issue with using the infrared sensor is the range, the medium range infrared sensor only has a range from 10 to 80 cm. Therefor, this sensor could become a very limiting factor as to how far the target can be placed from the cannon.

Camera tracking

Testing the camera will consist of several parts, these parts will heavily rely on the technical specifications of the camera. Therefor, the first part of this section will be on the technical specifications of the camera and the needed software.

The camera runs at 30 fps and with a resolution of 88 x 144, this could potentialy become an issue since it is a very low resolution. But that depends on how large the object/objects being tracked are. It also has the ability to track up to 8 different objects.

The setup of the camera is quite simple and is done using a computer, here it is setup which color it is suposed to track. The process of choosing the color is done

Part II

Testing

Chapter 5

Projectile Physics

In order to accurately fire a projectile, there is a number of information which will be required. This information mainly consists of theorems relating to projectile motion and the trajectory of projectiles under the influence of earths gravity.

This chapter will be used to present, explain and discuss the physical phenomena which are relevant to the motion of the cannons projectile.

Projectile Motion

When fireing a projectile it is possible to determine its exact trajectory by using the formula for projectile motion. This formula can be used to map the projectiles relative location to the turret over a given time frame. This formula is split into two parts for calculating the x- and y-coordinate respectively. The formulas are as follows:

$$x = v_0 * cos(a) * t \tag{5.1}$$

$$y = v_0 * sin(a) * t - \frac{1}{2} * g * t^2 \tag{5.2}$$

The formulas use the following variables:

- v_0 (m/s) - Starting Velocity
- a $(degrees)$ - Angle of Fire
- g (m/s^2) - Gravitational Acceleration
- x (m) - distance
- t (s) - Time

In addition, the following variables are used in the following sections:

- y_0 (m) - Height of Fireing

- y (m) - height

It should be noted that the formulas Equation 5.1 and Equation 5.2 do not take into accound any effect air resistance could have on the projectiles trajectory. The topic of air resistance and the relevance thereof is discussed in section 5.3.

Calculating Distance

In order to calculate the distance at which the projectile will touch the groud we need to combine the expressions in order to express the horisontal distance **y** as a function of the height **x**. The first step is to take Equation 5.2 and express the time **t** as a function of the height **y**. This is done as the resulting expression can be substituted into Equation 5.1. Doing this gives the equation:

$$x = v * cos(a) * \frac{v * sin(a) + \sqrt{v^2 * sin(a)^2 - 2 * g * y}}{g} \tag{5.3}$$

As such Equation 5.3 can be used to calculate the distance the projectile can travel on any given shot.

Angle-Distance Relation

When fireing a projectile the turret has one primary variable it can use to aim namely the angle at which the projectile is fired. This variable is theoretically enough to accurately fire the turret, but it does present some problems when it comes to calculating the trajectory and the offset needed for targets in motion.

Given a constant initial velocity of the projectile, the distance travelled and air time is proportional to the fireing angle. This relation is shown in Equation 5.3 where **cos(a)** and **sin(a)** are used as coeffecients. The relation between fireing angle and travelled distance is exemplified in Figure 5.1.

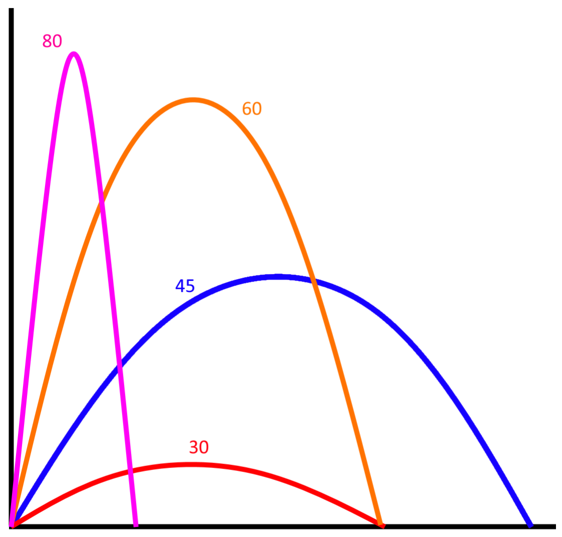

Figure 5.1: Distance relative to angle of fire.

Projectile Air Time

When it comes to hitting a target in motion we need another crucial information, namely how long a given projectile takes to reach its target. In subsection 5.1.1 we determined a formula for calculating the distance at which the projectile would hit the ground. As we have already determined the distance, we can simply use the formula described in Equation 5.1, namely:

$$x = v_0 * cos(a) * t \tag{5.4}$$

By isolating for the time in this formula, we can determine how long it takes for at given projectile to reach the target. This results in the following formula:

$$t = \frac{x}{v_0 * cos(a)} \tag{5.5}$$

Fireing Offset

As described in the requirements in **??** the cannon needs to calculate an offset to be able to hit a target in motion. In order to calculate this we will need ...

This can be calculated using Equation 5.3 which was derived in section 5.1.

Air Resistance

For the purpose of calculating the trajectory of the projectile in **??** a formula has been used which does not account for the effect of air resistance. This section will be used to discuss why this is the case, and to argue why the effect of air resistance is neglible in the case of fireing the chosen projectile.

In cases where a projectile is fired at high speeds, and is travelling for a longer period of time it is important to factor in the effect of air resistance, as this can have a significant effect on the trajectory of te projectile. On the other hand when we have relatively slow moving projectiles which are only mid air for a short duration, the effect of air resistance becomes much less important.

Another reason for using the simplified formula which disregards air resistance is that the Lego Nxt has limited processing power, and that the turret has strict deadlines. This leads to a situation where the speed of calculation is more important than the unnecessary precision of factoring in air resistance.

Target Requirements

Fireing Offset

As described in the requirements in **??** the cannon needs to calculate an offset to be able to hit a target in motion. In order to calculate this we will need ...

This can be calculated using Equation 5.3 which was derived in section 5.1.

Bibliography

[1] Frank Vahid and Tony Givargis. Embedded system design: a unified hardware/software approach. *Department of Computer Science and Engineering University of California*, 1999.

There are 1 places that needs fixing page 6