Lab 4

When barcodes attack!

Table of Contents

Aim

The aim of the laboratory is to solve a moderately large concrete software problem – having your robot read giant barcodes by driving over them – using the design and programming skills you have learned so far during the course.

Barcodes – An Overview

Barcodes encode their information according to the width of their bars. Both the black bars and the white ones encode information. The important thing is the relative width of these bars.

A common format is Code 39. It is widely used for inventory keeping and other custom applications because it is easy to implement and represents information flexibly. The coding system used on retail packaging (UPC) is different from Code 39.

A Code 39 character code is made up of five black bars separated by four white spaces. Exactly three of these elements (bars or spaces) will be wide, the others narrow. For example, the character `*` is represented by `NWNNWNWNN`, where `N` represents a narrow element and `W` represents a wide element. Each character code begins and ends with a black bar. The `*` character would therefore be encoded as:

The full list of Code 39 characters is available on the Code 39 Wikipedia page as well as the `code39.h` file provided to you.

Width ratio: The ratio between a wide and narrow element varies between 1.8:1 and 3.4:1 from one barcode to another. However, it is normally around 2.5:1. The ratio will be approximately consistent for a given barcode, but may vary slightly due to printing issues.

Each character's code is separated by a narrow white space. A complete barcode is formed by putting any number of characters between delimiting `*` characters. The string `TEST` would be encoded as:

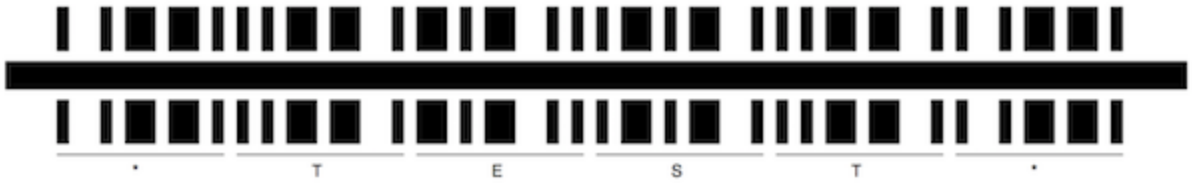
Procedure

You must design and implement a program that allows the robot to read a giant barcode of up to six characters plus delimiters.

For this lab, even more than previous labs, you need to think before you code. If have not designed a clear functional decomposition before you start, you won't likely be able to finish in a reasonable time.

You must use a structured design approach, including functional decomposition and demonstrate everything with a structure diagram. You should design each function on paper using either a flowchart or pseudo-code before attempting its implementation. Use your structure diagram and flowcharts or pseudo-code as a design tool. The instructor will ask you to see your structure diagram and flowcharts or pseudo-code before helping you debug. Your structure diagram must be submitted with your laboratory.

To help your robot stay aligned with the barcode, the codes have been printed on either side of a guide line, as shown below.



You should use the three middle sensors follow the line and the two on the far right and left to read the barcode. You may reuse your line-following code from lab 3.

Requirements

The robot must do the following:

1. When run, your robot must display an introductory screen similar to figure 1 of lab 3, with appropriate changes. You may reuse your code from lab 3.
2. When the B button is pressed, the robot must perform a calibration routine to allow the line detectors to properly differentiate between black and white.
3. After calibration, the robot must display `Ready` and wait.
4. The robot should now be placed in front of the barcode. When the B button is pressed, the robot must advance until it successfully reads the barcode or detects an error. To help with debugging, the robot must:
   a. play a low note every time it reaches the start of a new character; and
   b. play a high note every time it reads a wide code element.
5. When the barcode has been read in its entirety, *i.e.* when the end delimiter has been read, the robot must stop and display the characters read, excluding the delimiters, on the display.

6. Your robot must detect and handle errors. When an error is detected, the robot must stop and display the characters read up to that point on the display, and the name of the error below that. The following errors must be handled; you may also handle others.
   a. `Bad Code`: either more or fewer than three wide elements were detected in a character's code.
   b. `Too Long`: 8 character codes were read, but the last character was not the delimiter.
   c. `Off End`: The end of the centre line was detected before the second delimiter character was encountered.

Resources Provided

A representation of the Code 39 encoding table is provided to you in the `code39.h` file. Look at `code39.h` and make sure you understand what it contains. You must use the data provided by `code39.h` in your program and you must not modify `code39.h` in any way.

Several test barcodes are available in the laboratory with the encoded characters written below the code. Valid and invalid codes are provided.

Assumptions

You may make the assumptions listed below. If you want to make any other significant assumptions in your design, you must obtain permission from one of the instructors and document the assumption in your report. Valid assumptions are:

- the robot starts on a guide line at an unknown distance from the barcode;
- the space on either side of the guide line at the starting position will be white;
- the robot will be oriented so that it encounters the barcode as it moves forward;
- the first black element read by the robot is the start of the barcode;
- the first character encountered will be a delimiter; and
- the width ratio between a wide and narrow bar will be as specified above.

Constraints

You may **not** assume that:

- The barcode that the robot reads is the same size as the test barcode. Barcodes can be printed at a larger or smaller scale than the test code.
- The barcode that the robot reads has the same ratio between wide and narrow bars as the test code. You must be able to read codes with ranges that meet the standard specified above.

Development process

You should take an incremental approach. Deconstruct the program into its smallest interesting pieces and solve them one by one. For example, one of the first things your program must do is determine what the wide and narrow bars look like in the current barcode. An initial approach could be to write a program that ignores the starting white space, reads the first 9 bars and calculates the threshold. View the values on the LCD screen so you are confident your robot can tell the difference between a wide and narrow bar.

You should move your robot relatively slowly, at least at the beginning of your development. You can try increasing the speed when your detection seems to be working well.

You need to decide how you will convert a sequence from barcode to character using `code39.h`. Some possible approaches are:

- Go through the array one element at a time looking for a matching pattern.
- Starting from the character array, create a one-dimensional array of character strings where the characters for narrow and wide form a complete character string such as `"NNNWWNWNN"` (`N` = narrow and `W` = wide) for the character `'0'`. You must have the null character (`'\0'`) at the end of each character string. You could then use a function like `strcmp()` to find the matching string.
- Write your own `strncmp()` function to search for `n` characters and stop without needing the null character, then use it on the array provided to you.
- Create a new array of integers where you treat each `N` as a `0` and each `W` as a `1` and use these values to compare.
- Create an array of structures that can contain the character, the value, and a key such as the combined character string or binary string.

No matter which approach you choose, you should develop it on paper first to ensure it will work.

Experience shows that the most difficult part of the laboratory is managing the space between characters. Make sure you understand how reading should be done.

Submission

Your submission must include:

1. All `.cpp` and `.h` files that make up your implementation. Do not include the complete CLion project, just the source files.
2. A report, as a single PDF document, explaining your design. This must contain:
   a. an introduction that describes the purpose of the laboratory in your own words;
   b. a detailed description of your design strategy;

c. a clean structure diagram of your program (we suggest using https://draw.io or Microsoft Visio); and

d. your source code, as an appendix. See the lab 2 instructions for tips on exporting your code to PDF.

Submit these as individual files (not zipped together) to the Moodle assignment for this lab.

Follow good coding practices and document your code well. Refer to the course ***C coding style***.