CSE2425 Robot Lab

NetID: gbot
Student Number: 5092094

Introduction

The following report describes all algorithms used in the robot assignments. The appendix covers how each sensor or component was used. Note that this was written with the components provided for the robot lab in mind. Different components may need to be interacted with in another way.

Timer settings

No timer settings were changed and/or set explicitly for the microcontroller[1] used. All code makes use of the built-in $delay(unsigned\ long)$ and $delayMicroseconds(unsigned\ int)$ functions provided by the Arduino library.

Assignment 2: Obstacle detection

To ensure the robot does not drive into any objects, it must detect objects in front of the robot and then stop. This can be achieved by using the front ultrasound sensor to calculate the distance between it and the nearest object in a straight line. Using this distance, the robot was programmed to either go forward or to stop if there is an object nearby, specifically when $d < 15cm$.
To avoid halting the main program loop when waiting for an echo, interrupts are used. The interrupt service routine is called when the echo pin[2] changes value. It will keep track of when the pin goes high (start of the pulse), and when the pin goes low (end of the pulse), then, once it goes low the total duration can be calculated, which in turn can be used to calculate distance.

Assignment 3: Line following

For the robot to be able to follow lines (straight lines, bends and hard 90° turns only), two infrared sensors[3] are used. These are located on either side of the line to be followed and detect whether the ground below them is black (like the line) or not.
If both sensors are not above black, the robot assumes the line is in between the sensors and just drives forward. If one of the sensors is above black, the robot turns until the other sensor is above black and vice versa. This may not be the most efficient solution, but it does consistently follow a line with hard 90° turns accurately.
An alternative solution would be to turn the robot until the sensor detecting black no longer detects black, and then drive forward again until either sensor detects black again, but in practice the robot had a hard time following the hard 90° turns (regular turns worked fine).

[1] STM32F103C8T6 microcontroller
[2] See Appendix: Interacting with components: Ultrasound sensors
[3] See Appendix: Interacting with components: Infrared sensors

Assignment 4: Platoon forming

To make the robot follow a moving obstacle (like another robot), a program similar to the obstacle detection algorithm[4] is used.
On each iteration, the distance between the robot and the obstacle is checked, and compared to the distance from the previous iteration.
The goal is to keep the robot's inter-distance between 10 to $20cm$, so when the distance gets near the lower or upper bound of this range the robot goes backwards or forwards at maximum speed respectively to try to stay within range. Note that if the object the robot is following moves faster than the robot's maximum speed, it is impossible to keep up and therefore it will not keep the intended inter-distance.
While in range, the robot's speed is slightly adjusted by increasing or decreasing it a bit depending on if the robot is losing or winning distance. These adjustments are not clearly noticeable in practice.

Assignment 5: Fixed-distance traveling

To make the robot drive a given distance (in a straight line), the speed encoders[5] will be used. Based on the circumference of the wheels and the amount of slots of the caster wheel, the traveled distance relates directly to the pulses read from the speed encoders. This means the robot has to drive until it has received the required amount of pulses. A slight error correction is used to take braking distance into account (≈ 4 pulses).
Pulses are counted on one wheel only, because the robot drives in a straight line (theoretically), meaning that both motors should have the same speed.
The speed encoder produces a very noisy signal. To make it usable, the algorithm polls the encoder signal on every iteration. Then, a moving average[6] is calculated to smoothen the signal. If the moving average passes a certain threshold, it is counted as a pulse. Note that the algorithm checks if it passes the threshold, not if it is above it, meaning that averages of future consecutive iterations that are also above the threshold are not counted as a pulse.
In addition to counting pulses, a line following algorithm is used to make sure the robot drives in a straight line.
In a perfect world, the robot drives in a straight line when both motors receive the same power. In practice this is not the case: the robot will turn slightly making measuring the driven distance difficult. A line following algorithm is used to make the robot stay in a straight line. This algorithm differs from the algorithm used in assignment 3. It only slightly adjusts the trajectory by making one motor turn a little faster (and thus slightly turning the robot) to try to make sure it stays on a straight line. This algorithm is only designed to support straight lines and tries to zigzag as little as possible to drive in a straight line while measuring the distance driven as accurately as possible..

[4] See Assignment 2: Obstacle detection
[5] See Appendix: Interacting with components: Speed encoders
[6] See Appendix: Moving average

Appendix

Important components[7]

- HC-SR04 ultrasound ranging sensor
- L9110 H-Bridge
- TCRT5000 Infrared reflective sensor
- B83609 speed encoder
- Motors with gearbox
- Caster wheel
- Red LEDs

Interacting with components

All algorithms covered in this report use the same driver file[8] to have easy access to the sensors, LEDs and motors without having to do I/O[9] manually.

LED control

The LEDs are connected to ground and to the microcontroller. To turn the LEDs on or off, the microcontroller's pin needs to be set to high or low respectively.

Infrared sensors

The infrared sensors measure how much light is reflected off a surface. They output a continuous analog signal. The sensor's sensitivity can be changed by using the potentiometer on the component, but this was not really needed. The read values are arbitrary, so depending on the type of reflective surface, thresholds need to be adjusted manually.

Ultrasound sensors

The ultrasound sensors measure time it takes for a sound wave to reflect on something and travel back to the sensor. The outcome can be used to calculate distance between the sensor and the reflecting object.
These sensors must be manually activated with a $10ms$ pulse on the trigger pin. Then, the echo pin (output) stays high until a response is received. The duration of this pulse can be converted to a distance. As per the component's documentation, a constant $58ms\ cm^{-1}$ is used for this conversion. It is necessary to make the trigger pin low before activation by setting it to low and then waiting a few microseconds.

[7] This list only includes components covered in the *Interacting with components* section.
[8] This is the *control.ino* file in the delivery.
[9] Input/Output: Interacting with the sensors and other components with the microcontroller in this case.

Motor control

To control the motors, an H-bridge is used. This is a circuit that switches the polarity of the voltage applied. In this case, it is used to be able to make the motor turn both directions. The H-bridge uses two control signals per motor. Setting one of these signals to high makes the motor turn one way, making the other signal high and the first one low makes it turn the other way. Both signals low of course means the motor will not turn. Additionally, it supports PWM[10] signals. This allows for controlling the power of the motor and therefore its speed more precisely.

Speed encoders

The speed encoders use a photoelectric sensor to detect if there's something blocking the optical channel. This is what the encoders produce as output. For the robot, a wheel with slots is used with the sensor. This can then be used to calculate how much said wheel has turned. On the robot, the slotted wheels are directly connected to the motor axle with the driving wheel, meaning that one full turn of the wheel corresponds to a full turn of the slotted wheel, and thus 20 pulses as the caster wheels used have 20 slots. The amount of pulses correlates to how much a wheel has rotated and the pulse frequency correlates to the robot's speed.
The wheels used have a circumference of about $21cm$, meaning that the pulses relate to distance driven: $\frac{20}{21}\,pulses\ cm^{-1} \approx 95\,pulses\ m^{-1}$.

Moving average

A moving average is the average value of the most recent x entries. It is calculated by keeping track of the past x entries and dividing their sum by x. To speed up these calculations, the summed value is saved. When a new entry gets added, its value is then added to the stored sum and the oldest entry is subtracted from it. This means it is not necessary to loop over all entries when calculating the sum on each iteration.

[10] Pulse-width modulation: A method to vary the average power delivered