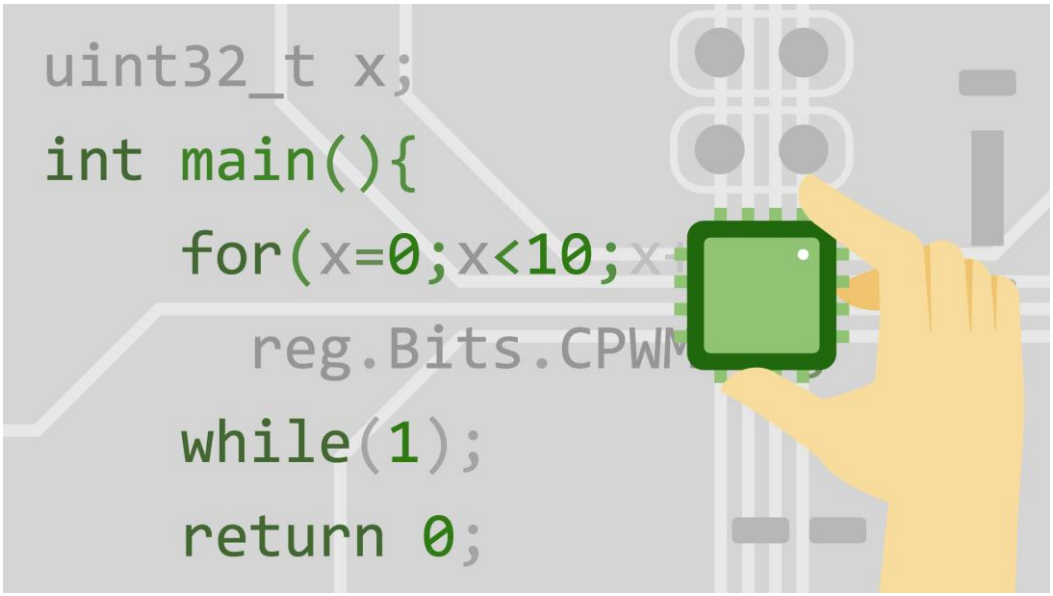
Elective C - embedded systems exam project

Désirée Schüller, Frederik Lundbeck Jørgensen



Which problem we solved

We chose to solve the second assignment the one-armed bandit.
The reasoning behind this was that we already tried our hand at the one-armed bandit last mandatory assignment, although that program didn't need to run and be controlled from the Arduino UNO.

What we planned to do

The project started by us defining which external functionalities/hardware should be involved with the game. Here's that list of features:

- A button that interacts with the game. Such as navigating the game menu and making the slot machine wheels spin.
- Blinking LED's when certain events happens in-game e.g. wheels are spun, price is won, game has ended etc.

Other than external requirements we also planned some internal ones:

- Clean and readable code, meaning commenting almost every line of code.
- Function & variable naming with no cryptic abbreviations.
- An object-oriented approach. Structs for things we can easily think of as objects e.g. the slot machine, the wheels, the game session etc.
- Separating said objects and their functionalities to their own .c and .h files.
- Using dynamic memory allocation by using the malloc function.

What we ended up doing

We implemented the one-armed bandit game into the Arduino UNO that can be played using a button. LED's are also mounted to the breadboard that lights up when certain events happens in-game.
The game can be played with two modes: Button mode which means you need to press the button to make the slot machine wheels spin. Auto-spin mode which

makes the program spin for you until you run out of credits.
A small game menu was also created with two of the three choices being selecting either of said modes.
We also created a statistic overview of the session at the end of the game. It displays data such as total amount of spins, highest winning price, longest win streak etc.
In hindsight we both agree with the conclusion that we've written maintainable and readable code.

One of the notable problems early in the project was the increasing dynamic memory we took up on the Arduino. With half the game done we had already used 80% of the RAM and it seemed to be all the text displayed in-game that was taking up the space. Then we found what seemed to be a magic bullet in compressing text data on the Arduino, Flash memory.
Simply by wrapping our static text (not changing) in the macro F() the string data would be saved into flash memory and thereby reducing our 80% usage of RAM down to 25%, amazing! 😊

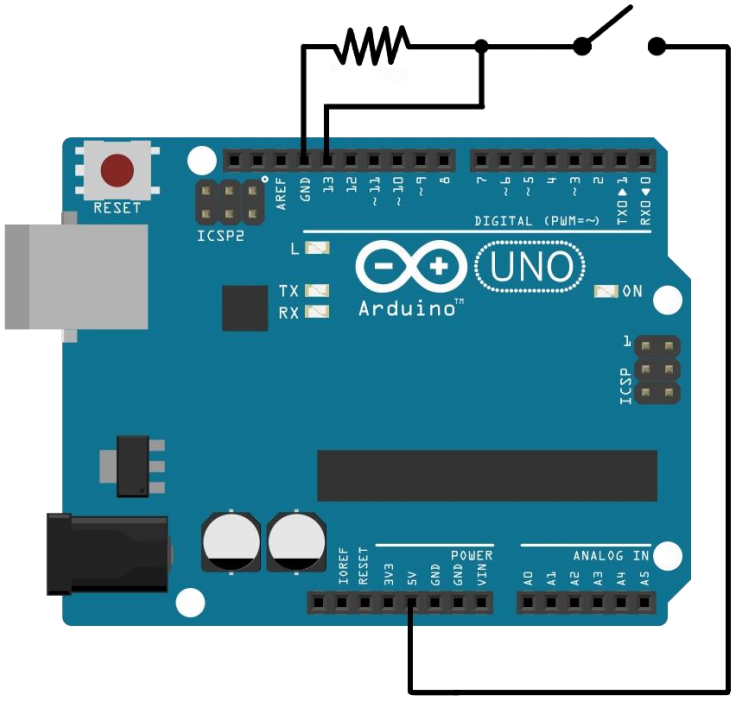
Important prerequisites for running the game

HARDWARE PREREQUISITES

1. An Arduino UNO. We do not recommend trying this on anything else than the UNO as we haven't yet tested the program on other microcontrollers.

2. Make sure you have a breadboard with a button circuit connected to

your Arduino uno (as it is required in order to play). Your button circuit should draw power from the 5V pin and needs to be connected with a jumper wire to pin 13.



SOFTWARE PREREQUISITES

You should only use this program through the serial monitor provided by the Arduino IDE and NOT other terminals that support serial connections like PuTTY etc.

Your Arduino IDE serial monitor should have the following options set in order to display the program correctly:

Auto scroll:	ON
Show timestamp:	OFF
Input:	No line ending
Baud:	9600