# 16 Automated hacking of the electronic combination lock

**Activity: Work the following exercises as directed by your tutor:**

## Purpose of this Activity

This activity will cover the use of micro-processor automation for finding a pre-programmed lock combination combining the use of digital input and output ports.

## Learning Outcomes

To be able to automate functions.

To be able to use hardware I/O to output digital signals in a pre-determined sequence and depending on input signal states.

* Concepts (in C): Loops, I/O control, Timing

## Activities

The three-digit electronic lock implemented previously can be hacked manually by trying out all combinations from 000 to 999 by hand. This is fairly time-consuming. A faster way is to use a processor that can try these combinations for us in a shorter time window.

## TASK 1:

Write a new C code file that iterates through all combinations of 000 to 999 in sequential order, i.e. 000, 001, 002, …, 009, 010, 011, … 999 and prints these on the screen.

Use three nested for-loops to achieve this. You may want to use variables named h, t, u for digits representing **h**undred, **t**ens, and **u**nits.

Here's a reminder of the for-loop syntax:

```
for ( <start value>; <while condition>; <value change> )
{
      …
}
```

For example:

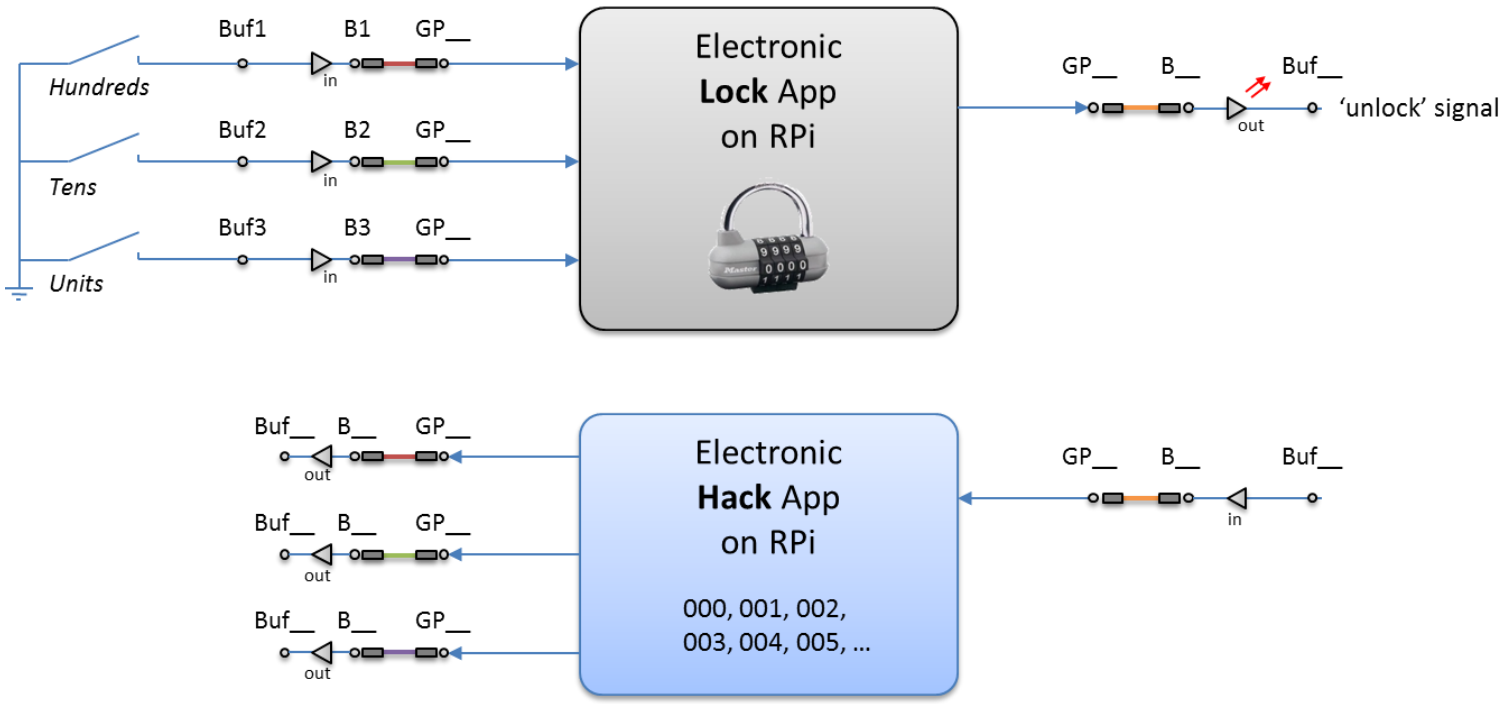
```
int u;

for ( u=0; u<=9; u++ )
{
      printf("%d \n", u);
}
```

Test the final code to ensure that the output as expected.

**TASK 2:**

Draw the necessary hardware connections to connect a hacking device to the lock.

Note: It will need three output lines connecting to the 'buttons' and one input line checking if the lock's output line has changed.




**TASK 3:**

Save the code from Task 1 into a new file with the name 'hacklock.c' and extend it with the necessary code and functions to implement the functions of the bottom block in Task 2:

1) Prompt the user which connections to make on the Gertboard.
2) Initialise the hardware correctly (setup_io, input and output configurations.
3) Run through all nested loops to generate the three digit codes.
4) Implement the required output signal line changes for simulating button presses that will connect to B1, B2, and B3.
5) Implement code that checks if the unlock signal has been activated. Stop the loop if this is the case and print out the unlocking combination.

Test your program.

**TASK 4:**

Connect up the wires between the hacking hardware and the lock hardware.

Run both the **lockApp** and the **hackApp** in parallel in two terminals.

Test the code and see if it works. What could be an issue here?

→ Insert delays where required for the code to stop at the correct digit combination.