Defuse the Bomb

A CSC 102 Project

Team: Rohan Khanal, Ricardo Almeida, Murat Yildiz

BOMB DEFUSAL MANUAL

Version 1

Verification Code: <enter your code>

**The Game**

This project is based on the game **Keep Talking and Nobody Explodes**[[1]](#footnote-1), a cooperative bomb defusing party game. As the game designers put it, “You’re alone in a room with a bomb. Your friends, the 'Experts', have the manual needed to defuse it. But there’s a catch: the Experts can’t see the bomb, so everyone will need to talk it out – fast! Put your puzzle-solving and communication skills to the test as you and your friends race to defuse bombs quickly before time runs out!”

Their version is a software game. Our version takes the idea and realizes it as a physical device with buttons, switches, and more! Although our version can be played just like theirs, players can interact with both the bomb and this document at the same time (i.e., players can both defuse the bomb and serve as the “Experts”, using this document to help disarm the phases).

The backend of our version of the game is a Raspberry Pi[[2]](#footnote-2) computer that combines a typical computer with the ability to interact with the outside world through sensors. The underlying software is written in Python[[3]](#footnote-3) and is the result of a final group-based project in CSC 102 (The Science of Computing II) in the Computer Science Program at the University of Tampa.

**Defusing Bombs**

The bomb will “explode” when its countdown reaches 0:00 or when too many strikes have occurred. You defuse the bomb by disarming all of its “phases” before the countdown expires.

**Phases**

The bomb has four phases, each of which must be disarmed to defuse the bomb. The phases can be disarmed in any order. Once a phase is disarmed, it becomes inactive, and changing it doesn't affect the bomb. Instructions for disarming the phases are provided in this document.

**Strikes**

A mistake in disarming a phase results in a strike. Get too many strikes, and the bomb “explodes. A sound will be played each time the user makes a mistake and the available stike counter is decreased by 1. The user fails if he exhausts all the 5 strikes and sound will be played for failure

**Information**

A different version of the bomb is randomly presented each time it is “booted”. There are 6,720 unique versions of the bomb with a whopping 1,176,000 possible variations!

Disarming some phases will require specific information about the bomb. Pay close attention to the “bootup” text on the bomb's screen.

A red and black device with a metal tube

Description automatically generatedRegarding the Toggles

It's so tempting to just toggle the switches over and over with those bright red LEDs and cool switch covers that you can flip. But one wrong toggle gets you one step closer to...BOOM!

The correct state of each toggle switch is based on the bomb's serial number.

You must first add the numeric digits in the serial number together to obtain a target value. Convert this value to a 4-digit binary number. Toggle the switches to represent the binary number.

Converting a number to binary (base 2) can be done by placing a 1 in the appropriate powers of two represented by the columns of the table below that, when added together, sum to the value. A 0 is placed in the remaining columns. The left-most digit of the binary number is known as the MSB (most significant bit), while the right-most digit is known as the LSB (least significant bit).

|  |  |  |  |
| --- | --- | --- | --- |
| 23 | 22 | 21 | 20 |
| 8 | 4 | 2 | 1 |
|  |  |  |  |

The left-most toggle switch represents the MSB, and the right-most toggle switch represents the LSB. The LED on a toggle switch lights up to represent a binary 1. Use the diagram below to assist you (which, by the way, represents the value 9).

A white and grey hexagon with a black background

Description automatically generatedA red circle in a white hexagon

Description automatically generatedA white and grey hexagon with a black background

Description automatically generatedA red circle in a white hexagon

Description automatically generated

The following model above shows an example of what 9 would look like with the toggles on in red representing as a 1 and the toggles that are switched off representing as 0.

Toggles are a binary so the user should know how to convert a normal digit number into its binary counter part, but how It works is the user is met with a math equation to solve which the answers could be from 1-15 as there are 4 toggles and the max is 1111 or 15 in binary so tell the users that when the toggles are off or down that represents a 0 and when the toggles are up that represents 1, for the description just tell them how to convert so if the answer is 10 you can teach them either the addition method or the division method whichever way you think that would be easier to the user

A circular logo with different colored circles

Description automatically generatedRegarding the Button

The button behaves in unpredictable ways. Follow the instructions below closely to avoid a strike!

At some point, you will need to press the button. However, releasing it is the hard part. The button has a lighted ring around it that can be red, green, or blue.

Release the button according to the following instructions:

|  |  |
| --- | --- |
| **Button color** | **Release instructions** |
| Red | Release the button at any time. |
| Green | Release the button when the first numeric digit found in the bomb’s serial number appears anywhere in the seconds of the countdown timer. |
| Blue | Release the button when the last numeric digit found in the bomb’s serial number appears anywhere in the seconds of the countdown timer. |

A screenshot of a phone number

Description automatically generatedRegarding the Keypad

Ooooh, an encrypted phase! Press the correct keys on the keypad carefully to avoid a strike. Try to avoid calling the “operator”.

Important information about the keypad is provided in the bomb's “bootup” text.

The correct combination can be determined by first decrypting a keyword with a key using an alphabetic substitution cipher – and then looking up the result in the table below to obtain a passphrase.

A substitution cipher with a numeric key represents a rotation of the alphabet. For a key of 5, for example, the alphabet is shifted five places such that A becomes F, B becomes G, …, and Z becomes E. Using this shift, for example, the word “THEY” encrypts to “YMJD”. Decrypting “YMJD” back to “THEY” is merely doing the reverse, effectively “subtracting” five positions from each letter.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Keyword** | **Phrases** |  | **Keyword** | **Phrases** |
| MIDWAY | FEIGN | RECALL | CLIMB |
| MIGHTY | CARVED | SYMBOL | LEAVE |
| REBORN | TRICK | SYSTEM | FOXES |
| BADGER | RIVER | WIDELY | BOUND |
| FIELDS | CYCLE | WINGED | YACHT |
| CANOPY | THROW | FIERCE | ALOOF |
| CABLES | SPINY | IMMUNE | STOLE |
| BANDIT | FADED | IMPACT | TOADY |

To enter the passphrase correctly, you must enter its numeric combination on the keypad. To do so, press each button on the keypad with the required letter only once.

This is the hard part where I would recommend more than one person do it because it will be the part that will take up the most amount of time. For this part the users will have to do a complex binary multiplication equation where when they find the answer in binary they will then have to convert that answer into a normal digit number and type it into the keypad. The # key is the delete key for if they type in a wrong digit and the \* is the enter key for when they are ready to submit their answer. For this part you’ll need to show the user how to convert from binary back to normal digits I would recommend making it easy on the user and showing them how binary is base 2 and each number for the base so like for example [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1048…] and so on and how they have to add the numbers together so their answer is 1001101 obviously the array in my example is the other way around and now from left to right but right to left they would add 1 + 4 + 8 + 64 to get there answer as a normal digit and input it into the keypad phase.

*A group of colored lines

Description automatically generated*Regarding the Wires

*Which wires should you “cut”? One wrong “snip” leads you one step closer to an “explosion”!*

The correct wires to “cut” is based on the color of the button and, in some cases, the bomb's serial number.

The wires are labeled as follows, depending on their orientation on your bomb:

A close-up of wires

Description automatically generatedA close-up of a wire

Description automatically generated

Note that the actual color of the wires doesn't matter, and the color of the wires on your bomb may be different than those in this document.

“Cut” the wires based on the color of the button according to the following instructions:

|  |  |
| --- | --- |
| **Button color** | **“Cut” instructions** |
| Red | The first three letters in the serial number represent the wires that must remain connected. The rest of the wires must be “cut”. |
| Green | “Cut” the wires labeled B and D. The rest of the wires must remain connected. |
| Blue | “Cut” all wires except those labeled B, C, and D. |

For the wires it’ll be a multiple choice question that the answer will be what wire the user must “cut” or pull out, the answer choices will be A B C D E which is correlated to the wires from left to right *THE COLORS OF THE WIRES DON’T MATTER JUST THE ORDER THEY ARE IN* the left most wire is A and it goes in alphabetical order to the right most wire being E. most of the questions are computer science related besides one which is for the university of Tampa. For this part the user will be met with a question displayed on the screen and when they answer it if wrong the bomb will blow up but if right it will move on to a next question where they’ll again have the same multiple choice -1 because one of the wires is now gone, there will be a total of 3 questions for this part for the user to answer for them to successfully disarm the bomb, I asked Dr K and for this part you can tell the user they can just look up the answer if needed unless they know it cause the questions are common

1. <https://keeptalkinggame.com/> [↑](#footnote-ref-1)
2. <https://www.raspberrypi.com/> [↑](#footnote-ref-2)
3. <https://www.python.org/> [↑](#footnote-ref-3)