**2TA4 Lab 5: Control of a stepper motor**

**Individual Lab Report**

**Alexander Bartella - 400308868**

1. The angular resolution of the motor is given by

$$Angular\ Resolution = \frac{360°}{\#\ of\ steps} = \frac{360°}{\#\ of\ phases * \#\ of\ poles}$$

In our case, the number of steps is known to be 48, therefore:

$$Angular\ Resolution = \frac{360°}{48\ steps} = 7.5°$$

2. The period that will be used to complete one full revolution is 35 seconds, since my student number is 400308868.

$$T = 68 - 33 = 35\ s$$

3. Since a full revolution will take 35 seconds, and there are 48 steps in a revolution, we can determine the time between any 2 steps of the motor.

a) We take twice as many steps in half-stepping, therefore:

Time between half-steps $= \frac{T}{2 * \#\ of\ steps} = \frac{35\ s}{96\ steps} = 0.3646$ s/half-step

b) For full-stepping:

$$time\ between\ steps = \frac{T}{\#\ of\ steps}$$
$$time\ between\ steps = \frac{35\ s}{48\ steps} = 0.7292\ s/step$$

4. From chapter 15, we know that:

$$f_{CK\_CNT} = \frac{f_{CK\_PSC}}{PSC + 1}$$

Where PSC is the prescaler value, $f_{CK\_PSC}$ is the frequency of bus APB1, and $f_{CK\_CNT}$ is the scaled frequency. In this case, the latter is the output frequency. Rearranging for the prescaler, we get:

$$PSC + 1 = \frac{f_{CK_{PSC}}}{f_{CK_{CNT}}}$$
$$PSC = \frac{f_{CK_{PSC}}}{f_{CK_{CNT}}} - 1$$

Solving for the prescaler value:

$$PSC = \frac{f_{APB1}}{f_{out}} - 1 = \frac{45\ MHz}{960\ Hz} - 1 = 46875 - 1$$
$$PSC = 46874$$

Therefore, the prescaler value needed is 3.

960 Hz*0.7292 s = ticks

Calculating the OCR value for full steps:

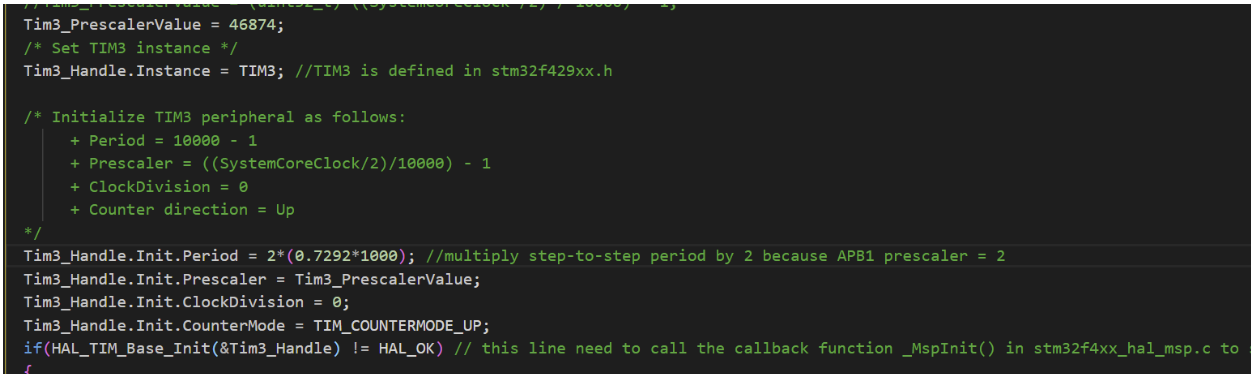
$$OCR_{full} = time\ between\ steps * f_{out} - 1$$
$$OCR_{full} = 0.7292\ s * 960\ Hz - 1$$
$$OCR_{full} = 699.03\ count/step$$

For Half-steps:

$$OCR_{half} = time\ between\ steps * f_{out} - 1$$
$$OCR_{half} = 0.3646 * f_{out} - 1$$
$$OCR_{half} = 349.016\ count/step$$

5.

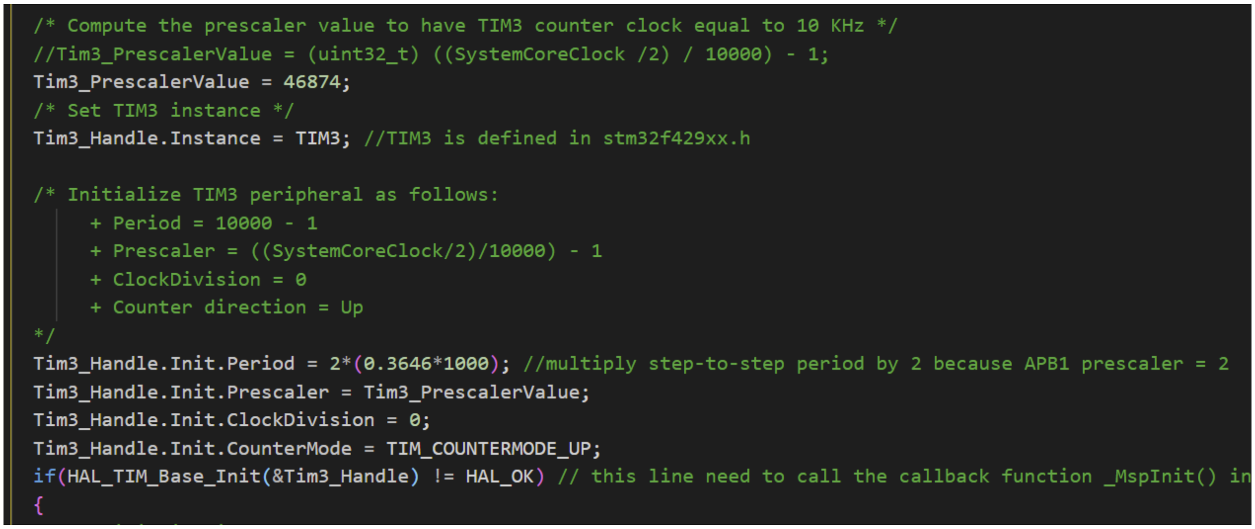
Full stepping config:



```
//Tim3_PrescalerValue = (uint32_t) ((SystemCoreClock /2) / 10000) - 1;
Tim3_PrescalerValue = 46874;
/* Set TIM3 instance */
Tim3_Handle.Instance = TIM3; //TIM3 is defined in stm32f429xx.h

/* Initialize TIM3 peripheral as follows:
     + Period = 10000 - 1
     + Prescaler = ((SystemCoreClock/2)/10000) - 1
     + ClockDivision = 0
     + Counter direction = Up
*/
Tim3_Handle.Init.Period = 2*(0.7292*1000); //multiply step-to-step period by 2 because APB1 prescaler = 2
Tim3_Handle.Init.Prescaler = Tim3_PrescalerValue;
Tim3_Handle.Init.ClockDivision = 0;
Tim3_Handle.Init.CounterMode = TIM_COUNTERMODE_UP;
if(HAL_TIM_Base_Init(&Tim3_Handle) != HAL_OK) // this line need to call the callback function _MspInit() in stm32f4xx_hal_msp.c to
```

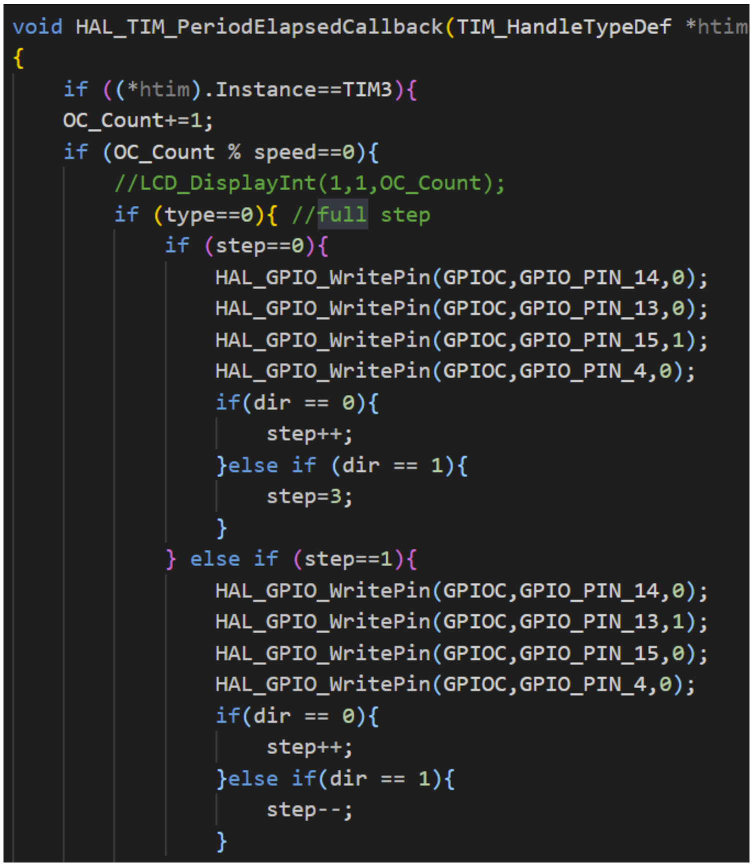
Half-stepping config:



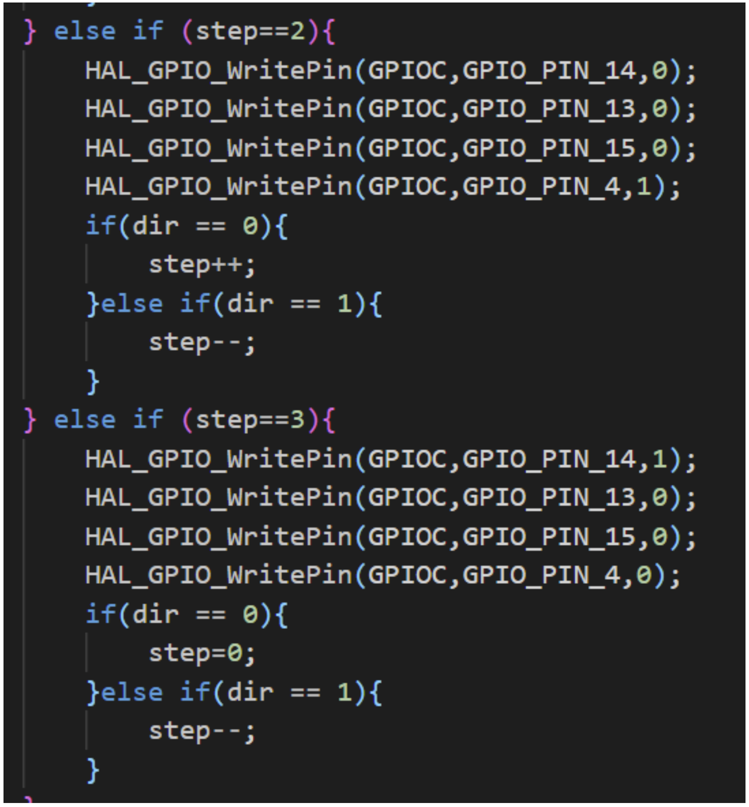
```
/* Compute the prescaler value to have TIM3 counter clock equal to 10 KHz */
//Tim3_PrescalerValue = (uint32_t) ((SystemCoreClock /2) / 10000) - 1;
Tim3_PrescalerValue = 46874;
/* Set TIM3 instance */
Tim3_Handle.Instance = TIM3; //TIM3 is defined in stm32f429xx.h

/* Initialize TIM3 peripheral as follows:
     + Period = 10000 - 1
     + Prescaler = ((SystemCoreClock/2)/10000) - 1
     + ClockDivision = 0
     + Counter direction = Up
*/
Tim3_Handle.Init.Period = 2*(0.3646*1000); //multiply step-to-step period by 2 because APB1 prescaler = 2
Tim3_Handle.Init.Prescaler = Tim3_PrescalerValue;
Tim3_Handle.Init.ClockDivision = 0;
Tim3_Handle.Init.CounterMode = TIM_COUNTERMODE_UP;
if(HAL_TIM_Base_Init(&Tim3_Handle) != HAL_OK) // this line need to call the callback function _MspInit() in
{
```

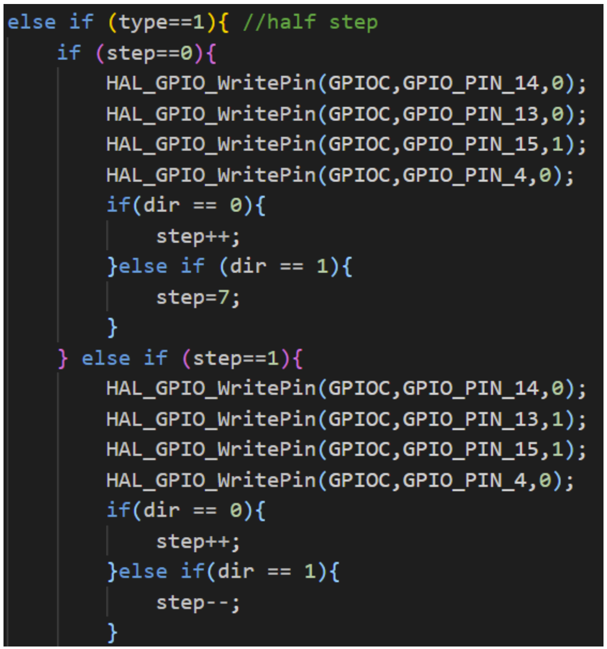
Motor implementations:



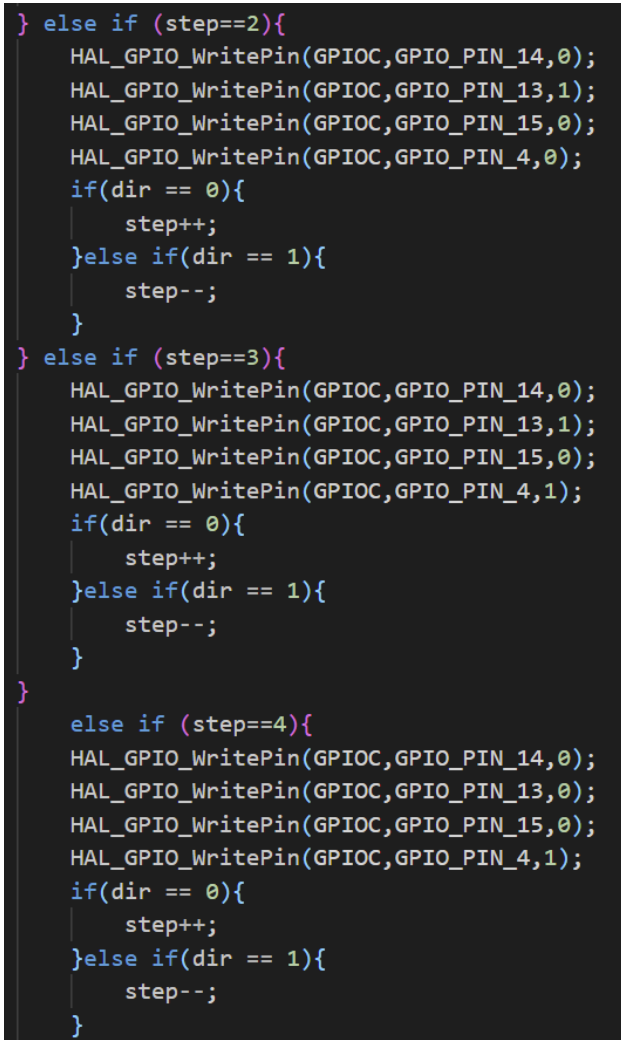
```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim
{
    if ((*htim).Instance==TIM3){
    OC_Count+=1;
    if (OC_Count % speed==0){
        //LCD_DisplayInt(1,1,OC_Count);
        if (type==0){ //full step
            if (step==0){
                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_14,0);
                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_13,0);
                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_15,1);
                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_4,0);
                if(dir == 0){
                    step++;
                }else if (dir == 1){
                    step=3;
                }
            } else if (step==1){
                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_14,0);
                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_13,1);
                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_15,0);
                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_4,0);
                if(dir == 0){
                    step++;
                }else if(dir == 1){
                    step--;
                }
```



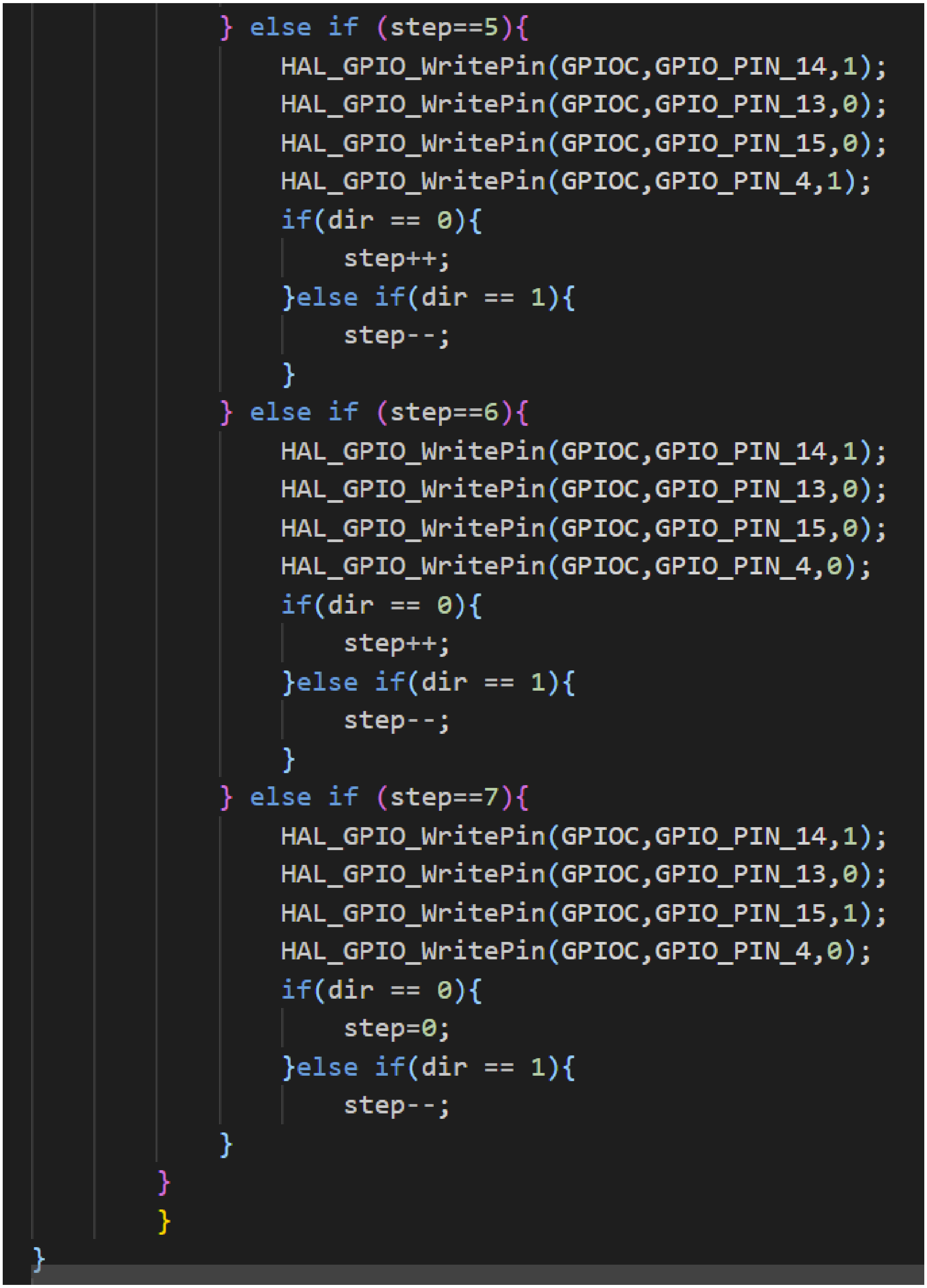
```
} else if (step==2){
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_14,0);
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_13,0);
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_15,0);
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_4,1);
    if(dir == 0){
        step++;
    }else if(dir == 1){
        step--;
    }
} else if (step==3){
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_14,1);
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_13,0);
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_15,0);
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_4,0);
    if(dir == 0){
        step=0;
    }else if(dir == 1){
        step--;
    }
```



```
else if (type==1){ //half step
    if (step==0){
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_14,0);
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_13,0);
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_15,1);
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_4,0);
        if(dir == 0){
            step++;
        }else if (dir == 1){
            step=7;
        }
    } else if (step==1){
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_14,0);
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_13,1);
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_15,1);
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_4,0);
        if(dir == 0){
            step++;
        }else if(dir == 1){
            step--;
        }
```



```
} else if (step==2){
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_14,0);
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_13,1);
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_15,0);
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_4,0);
    if(dir == 0){
        step++;
    }else if(dir == 1){
        step--;
    }
} else if (step==3){
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_14,0);
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_13,1);
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_15,0);
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_4,1);
    if(dir == 0){
        step++;
    }else if(dir == 1){
        step--;
    }
}
    else if (step==4){
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_14,0);
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_13,0);
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_15,0);
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_4,1);
    if(dir == 0){
        step++;
    }else if(dir == 1){
        step--;
    }
```



```
            } else if (step==5){
                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_14,1);
                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_13,0);
                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_15,0);
                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_4,1);
                if(dir == 0){
                    step++;
                }else if(dir == 1){
                    step--;
                }
            } else if (step==6){
                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_14,1);
                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_13,0);
                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_15,0);
                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_4,0);
                if(dir == 0){
                    step++;
                }else if(dir == 1){
                    step--;
                }
            } else if (step==7){
                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_14,1);
                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_13,0);
                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_15,1);
                HAL_GPIO_WritePin(GPIOC,GPIO_PIN_4,0);
                if(dir == 0){
                    step=0;
                }else if(dir == 1){
                    step--;
            }
        }
        }
}
```

see submitted code for more details.