```
#include "PC_FileIO.c"

void ReadFromFile(TFileHandle & fin, int* pill_1, int* pill_2, int* pill_3, float* wait_time_hrs);
float clipToBottom(float powerToMotor);
void openAndClose(int motor_num, int angle);
void Dispense(int container, int num_pills);
void moveElevationSensordist(int motor_power, float dist_cm);
bool checkPillLevel(int colour);
void refillPillAlert(bool pill_level);
void MedicationTimeAlert(int time);

const int PROPORTIONAL = 1.0/45.0;
const int ANGLE = 20;
const int NUM_ELEMENTS = 3;
task main()
{


  TFileHandle fin;
  bool fileOkay = openReadPC(fin, "pills_to_dispense.txt");

  displayString(2, "Press enter to begin");
  //Wait till Enter button is pressed
  while(!getButtonPress(buttonEnter))
   {}
  while(getButtonPress(buttonEnter))
   {}


  //File Input

  //declaring the arrays
  //Also, Populating the arrays manually:
  int pill_1[NUM_ELEMENTS];
  int pill_2[NUM_ELEMENTS];
  int pill_3[NUM_ELEMENTS];
  float wait_time_hrs[NUM_ELEMENTS];

  //Populates Arrays with info from files

  ReadFromFile(fin, pill_1, pill_2, pill_3, wait_time_hrs);




  //For loops to dispense pills

  for(int times = 0; times < 3; times++)
  {
   float time_in_Hrs = wait_time_hrs[times];  //We will multiply num hours by 10 seconds just for the purpos
es of the demo
   time1[T1] = 0;
   while(time1[T1] < time_in_Hrs * 10000)
   {}
   Dispense(1, pill_1[times]);
```

```
  Dispense(2, pill_2[times]);
  Dispense(3, pill_3[times]);
  MedicationTimeAlert(time_in_Hrs);
  //In this function it should wait until a button is pressed for the beeping sound to stop.
 }

 //booleans for refilling the containers
 bool refill_container_1 = false;
 bool refill_container_2 = false;
 bool refill_container_3 = false;

 //configuring the color sensor
 SensorType[S1] = sensorEV3_Color;
 SensorMode[S1] = modeEV3Color_Color;
 wait1Msec(100);
 //Moving the colour sensor up and down to check the level of pills
 refill_container_1 = checkPillLevel((colorRed));
 moveElevationSensordist(20, 13);
 refill_container_2 = checkPillLevel((colorYellow));
 wait1Msec(3000);
 moveElevationSensordist(20, 12);
 wait1Msec(3000);
 refill_container_3 = checkPillLevel((colorRed));;

 //Displaying messages and sounding alerts for containers that need to be refilled
 if(refill_container_1)
 {
  displayString(5, "refill pills in container 1");
  refillPillAlert(refill_container_1);
 }

 if(refill_container_2)
 {
  displayString(10, "refill pills in container 2");
  refillPillAlert(refill_container_2);
 }
 if(refill_container_3)
 {
  displayString(15, "refill pills in container 3");
  refillPillAlert(refill_container_3);
 }

 moveElevationSensordist(-20, -25);

 wait1Msec(10000);

}

void ReadFromFile(TFileHandle & fin, int* pill_1, int* pill_2, int* pill_3, float* wait_time_hrs)
{
int time_temp1 = 0, time_temp2 = 0, time_temp3 = 0;

readIntPC(fin, time_temp1);
readIntPC(fin, time_temp2);
readIntPC(fin, time_temp3);
```

```
wait_time_hrs[0] = time_temp1;
wait_time_hrs[1] = time_temp2;
wait_time_hrs[2] = time_temp3;

for(int count = 0; count<3; count++)
 {

 int temp1 = 0, temp2 = 0, temp3 = 0;
 readIntPC(fin, temp1);
 readIntPC(fin, temp2);
 readIntPC(fin, temp3);

 pill_1[count] = temp1;
 pill_2[count] = temp2;
 pill_3[count] = temp3;

 }
}

//function for pidf
float clipToBottom(float powerToMotor)
{
 if(powerToMotor < 0.1)
 {
  return 0.1;
 }
 else
 {
  return powerToMotor;
 }
}

//Open and close function
void openAndClose(int motor_num, int angle)
{
 nMotorEncoder(motor_num) = 0;
 while (abs(nMotorEncoder[motor_num]) < angle)
 {
  float error = abs(angle - nMotorEncoder[motor_num]);

  float motorPower = clipToBottom(error*PROPORTIONAL)*100 + 100;

  motor[motor_num] = motorPower;
 }
 motor[motor_num] = 0;
 nMotorEncoder(motor_num) = 0;
 while (abs(nMotorEncoder[motor_num])< angle)
 {
  float error = abs(angle - nMotorEncoder[motor_num]);

  float motorPower = clipToBottom(error*PROPORTIONAL)*100 + 100;

  motor[motor_num] = -motorPower;
 }
```

```
 motor[motor_num] = 0;
}

//dispense function
void Dispense(int container, int num_pills)
{
 int motor_name = 0;
 if(container == 1)
 {
  motor_name = motorA;
 }
 if(container == 2)
 {
  motor_name = motorB;
 }
 if(container == 3)
 {
  motor_name = motorC;
 }

 for(int num_time = 0; num_time < num_pills; num_time++)
 {
  openAndClose(motor_name, ANGLE);
  wait1Msec(500);
 }
}

//function that moves elevation sensor specific distance
void moveElevationSensordist(int motor_power, float dist_cm)
{
nMotorEncoder[motorD] = 0;
motor[motorD] = -motor_power;
float const RADIUS = 1.75;
int const ENC_LIMIT = dist_cm*180/(PI*RADIUS);
if(dist_cm > 0)
{
 while (abs(nMotorEncoder[motorD]) < ENC_LIMIT)
 {}
 motor[motorD] = 0;
}
else if( dist_cm < 0 )
{
 while (abs(nMotorEncoder[motorD]) < abs(ENC_LIMIT))
 {}
 motor[motorD] = 0;
}
return;
}

//Check pill level function
bool checkPillLevel(int colour)
{
 if(SensorValue[S1] == colour)
 {
  return false;
```

```
 }
 else
 {
  return true;
 }
}

void refillPillAlert(bool pill_level)
{

 if(pill_level == true)
 {
  while(!(getButtonPress(buttonEnter)))
  {
   playSoundFile("alarm");
  }
  while(getButtonPress(buttonEnter))
  {}

  clearSounds();
 }
 return;
}

void MedicationTimeAlert(int time)
{


 while(!getButtonPress(buttonEnter))
 {
  playSoundFile("pills_reminder_2");
 }
 while(getButtonPress(buttonEnter))
 {}
 clearSounds();


}
```