File: C:\1B_spring2025\ME101\Project\software\ME101PROJECT\mainProgram.c

```
/*
MOTOR A - DRIVES THE ROBOT
MOTOR B - MOVES SORTING ARM
MOTOR C - MOVES CASH BOX
*/

//GLOBAL VARIABLES HERE:
bool billsLeft=true;
bool resetHit = false;

//configure all sensors
void configureAllSensors()
{
SensorType[S1] = sensorEV3_Touch;
SensorType[S2] = sensorEV3_Ultrasonic;//distance to drive
SensorType[S3] = sensorEV3_Color;
wait1Msec(50);
SensorMode[S3] = modeEV3Color_Color;
wait1Msec(50);
SensorType[S4] = sensorEV3_Ultrasonic;//cash distance from top of pile
wait1Msec(50);

}



//hard stop and reset
void stopEverything()
{

  if(nMotorEncoder[motorC]>0)
  {
  motor[motorC]=-50;
  wait1Msec(900);
  motor[motorC]=0;
  }

  if(SensorValue[S2]<=55)

  {
  motor[motorA]=100;
  while(SensorValue[S2]<=55)
  {}
  motor[motorA]=0;

  }


    motor[motorB]=-100;
    wait1Msec(100);
    motor[motorB] = 0;




billsLeft=false;
resetHit = true;
```

```
}

//drive robot given distance
void driveRobot(int dist, int speed)
{
  nMotorEncoder[motorA]=0;
  clearTimer(T1);
  int timeout = 1750

  motor[motorA]=speed;
  while(SensorValue[S2]>dist&&time1(T1)<timeout)
  {
  if(SensorValue[S1]==1)
      {
      stopEverything();
      }
  }

  motor[motorA]=0;

}

//reset robot to origin using touch sensor
void resetRobot(int speed)
{
  motor[motorA]=speed;
  while(SensorValue[S2]<55)
  {
      if(SensorValue[S1]==1)
      {
      stopEverything();
      }
    }

  motor[motorA]=0;
}

//pickup/release bill, positive direction brings the arm down
void armDown(int direction, float armDist)
{
  nMotorEncoder[motorB] = 0;
  motor[motorB] = direction * 100;

  clearTimer(T1); // start a timer
  int timeout = 3000; // 3 seconds

  // Move until encoder reaches target based on direction
  if (direction > 0)
  {
    while(nMotorEncoder[motorB] < armDist * 175 && time1[T1] < timeout)
    {
      wait1Msec(10);
    }
  }
  else if (direction < 0)
  {
    while(nMotorEncoder[motorB] > -armDist * 175 && time1[T1] < timeout)
    {
      wait1Msec(10);
    }
  }

  motor[motorB] = 0;
}





//return value of bill
int getBillValue(int billColor)//DONE BUT DOESNT READ OUR BLUE PROPERLY
{
  int storedValues[6]={5,20,10,50,0,100};


  for(int i=2; i<=7; i++)
  {

    if(i!=6)
    {
      if(billColor==i)
      {
        return storedValues[i-2];
      }
    }
  }
  return 0;

}

//get bill value from color sensor
int getBillColor()//DONE
{
int colorNum=0;
colorNum=SensorValue[S3];
return colorNum;
}

//convert color integer to dist in cm
float getDist(int billColor)
{
float distCM=0;

  if(billColor == (int) colorBlue)
  distCM=42;

  if(billColor== (int) colorYellow)//yellow in place of purple
  distCM=32;

  if(billColor== (int) colorGreen)
  distCM=22;

  if(billColor== (int) colorRed)
  distCM=12;

  if(billColor== (int) colorBrown)
  distCM=2.5;
```

```
return distCM;
}

//check if cash box is positioned high enough and adjust if necessary
void confirmCashBoxPosition()
{
   if(SensorValue[S4]>15.5)
   {
     motor[motorC]=10;
     while(SensorValue[S4]>15.5)
     {}
     motor[motorC]=0;
   } if(SensorValue[S4]<15.5)
   {
      motor[motorC]=-10;
     while(SensorValue[S4]<15.5)
     {}
     motor[motorC]=0;
   }

}


task main()
{
configureAllSensors();

int totalValue=0;
int billColor, billCount = 0;
int testVal =0;


while(billsLeft)
{
billColor=getBillColor();

wait1Msec(500);
confirmCashBoxPosition();
wait1Msec(1000);


armDown(1,1);
wait1Msec(500);
armDown(-1,0.5);


driveRobot(getDist(billColor), -100);

armDown(-1,0.75);

wait1Msec(500);
resetRobot(50);


if(SensorValue[S3]==0||SensorValue[S3]==1)
{
  billsLeft=false;
}
totalValue+=getBillValue(billColor);
billCount++;
}

if(!resetHit)
{
motor[motorC]=-540;
wait1Msec(900);
motor[motorC]=0;
}

displayString(0,"total number of bills is  %d",billCount);
displayString(1,"total value is $%d",totalValue);
displayString(2,"distance is %d",testVal);
wait1Msec(10000);



}
```