**CG1112 Engineering Principles and Practices II**

**Week 9 Studio 2**

**Answer Book**

**Name: Lee Raiyan bin Zulkifli**

**Student ID (AxxxxxxY): A0183903A**

**Question 1a** (3 marks) / **Question 1b** (3 marks)

(You may find it easier to submit a single piece of code for both parts a and b together)

// Functions to be called by INT0 and INT1 ISRs.

void leftISR()

{

if(dir == FORWARD){

leftForwardTicks++;

forwardDist = (unsigned long) ((float) leftForwardTicks / COUNTS\_PER\_REV \* WHEEL\_CIRC);

}else if(dir == BACKWARD){

leftReverseTicks++;

reverseDist = (unsigned long) ((float) leftReverseTicks / COUNTS\_PER\_REV \* WHEEL\_CIRC);

}else if(dir == LEFT){

leftReverseTicksTurns++;

}else if(dir == RIGHT){

leftForwardTicksTurns++;

}

//leftRevs = leftTicks / COUNTS\_PER\_REV;

// We calculate forwardDist only in leftISR because we

// assume that the left and right wheels move at the same

// time.

//forwardDist = leftRevs \* WHEEL\_CIRC;

//Serial.print("LEFT: ");

// Serial.println(leftTicks);

}

void rightISR()

{

if(dir == FORWARD){

rightForwardTicks++;

}else if(dir == BACKWARD){

rightReverseTicks++;

}else if(dir == LEFT){

rightForwardTicksTurns++;

}else if(dir == RIGHT){

rightReverseTicksTurns++;

}

//rightRevs = rightTicks / COUNTS\_PER\_REV;

//Serial.print("RIGHT: ");

// Serial.println(rightTicks);

}

**Question 2** (2 marks)

The Pi passes two parameters for COMMAND\_FORWARD, COMMAND\_REVERSE: distance and speed.

The Pi passes two parameters for COMMAND\_TURN\_LEFT and COMMAND\_TURN\_RIGHT to the Arduino: angle and speed.

**Question 3** (2 marks)

**Question 4** (2 marks)

The function sendOK() sends a response back to the Pi to acknowledge that the packet has been successfully sent to the Arduino.

**Question 5** (2 marks)

case COMMAND\_TURN\_LEFT:

sendOK();

left((float) command->params[0], (float) command->params[1]);

break

**Question 6** (3 marks)

void sendStatus()

{

// Implement code to send back a packet containing key

// information like leftTicks, rightTicks, leftRevs, rightRevs

// forwardDist and reverseDist

// Use the params array to store this information, and set the

// packetType and command files accordingly, then use sendResponse

// to send out the packet. See sendMessage on how to use sendResponse.

//

TPacket statusPacket;

statusPacket.packetType=PACKET\_TYPE\_RESPONSE;

statusPacket.command = RESP\_STATUS;

statusPacket.params[0] = leftForwardTicks;

statusPacket.params[1] = rightForwardTicks;

statusPacket.params[2] = leftReverseTicks;

statusPacket.params[3] = rightReverseTicks;

statusPacket.params[4] = leftForwardTicksTurns;

statusPacket.params[5] = rightForwardTicksTurns;

statusPacket.params[6] = leftReverseTicksTurns;

statusPacket.params[7] = rightReverseTicksTurns;

statusPacket.params[8] = forwardDist;

statusPacket.params[9] = reverseDist;

sendResponse(&statusPacket);

}

**Question 7** (3 marks)

void handleCommand(TPacket \*command)

{

switch(command->command)

{

// For movement commands, param[0] = distance, param[1] = speed.

case COMMAND\_FORWARD:

sendOK();

forward((float) command->params[0], (float) command->params[1]);

break;

case COMMAND\_REVERSE:

sendOK();

reverse((float) command->params[0], (float) command->params[1]);

break;

case COMMAND\_TURN\_LEFT:

sendOK();

left((float) command->params[0], (float) command->params[1]);

break;

case COMMAND\_TURN\_RIGHT:

sendOK();

right((float) command->params[0], (float) command->params[1]);

break;

case COMMAND\_STOP:

sendOK();

stop();

break;

case COMMAND\_GET\_STATS:

sendOK();

sendStatus();

break;

case COMMAND\_CLEAR\_STATS:

sendOK();

clearOneCounter(command->params[0]);

break;

default:

sendBadCommand();

}

}

**TOTAL:** \_\_\_\_\_ / 20