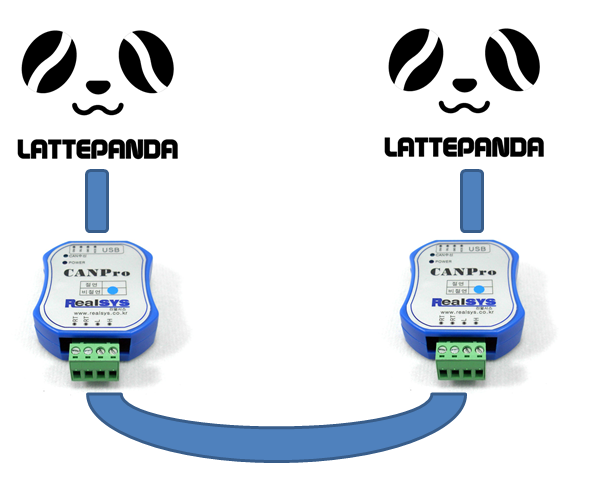
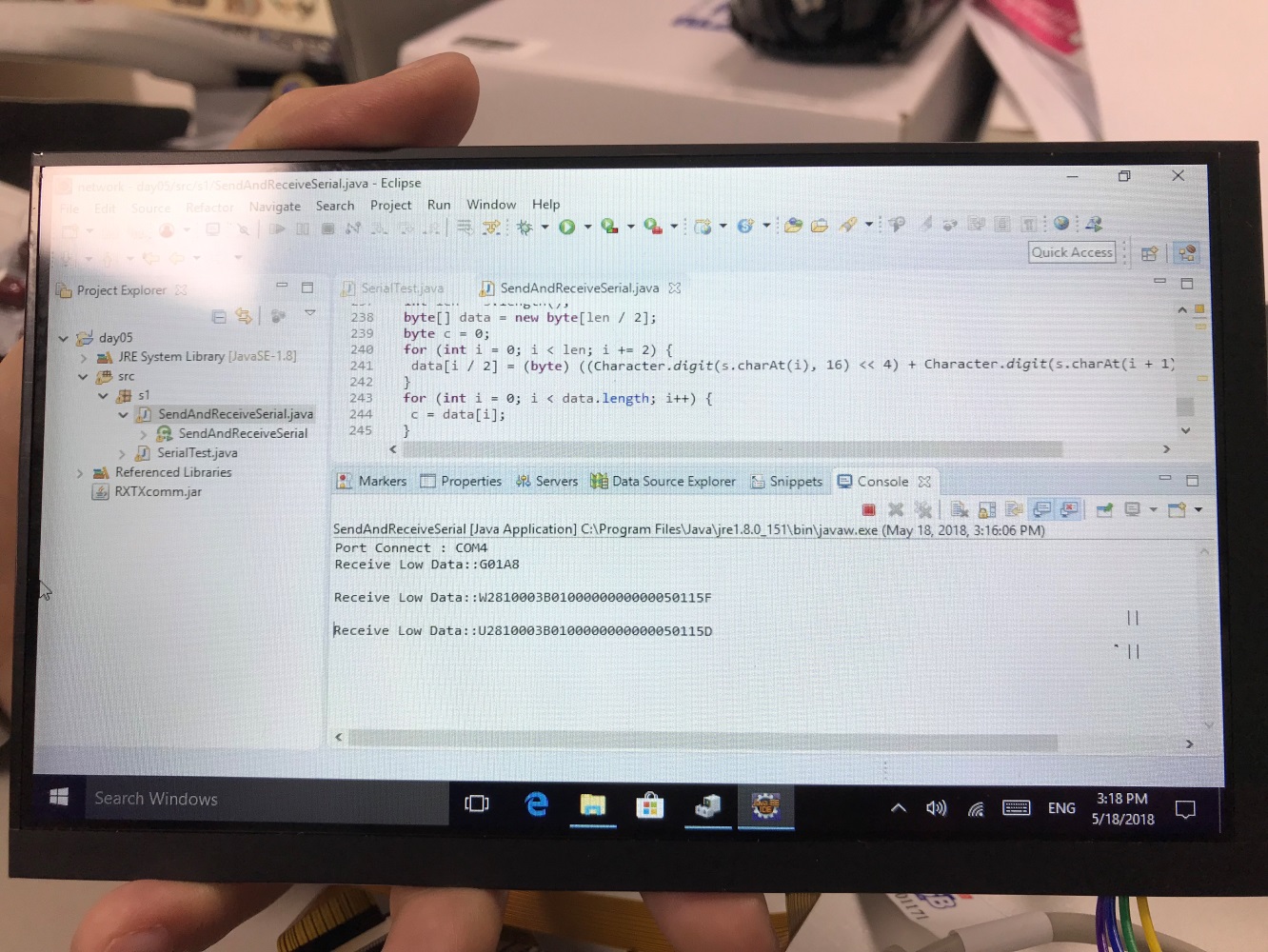
|  |  |  |  |
| --- | --- | --- | --- |
| 과정명 | 커넥티드 카를 위한 IoT 전문가 과정(NCS) | | |
| 교과목명 | 커넥티드 카 스마트 패드 및 사용자  컨트롤러 구축을 위한 안드로이드  프로그래밍 – 실습 프로젝트 | 훈련교사 | 이진만 |
| 평가일시 | 2018년 5월 11일 | 훈련생명 |  |
| 프로젝트 명 | CAN 통신 및 스마트패드 연동 보드개발을 위한 IoT 프로그램 | | |
| 개발기간 | 2018.05.04 ~ 2018.05.11 | 팀원명 | 백상우 |
| 개요 | * IOT 장비와 CAN 통신 연동 및 데이터 송수신 모듈 개발 * CAN 연동 모듈이 정상적으로 설치 되었는가 * CAN 연동 송신 모듈이 정상적으로 개발 되었는가 * CAN 연동 수신 모듈이 정상적으로 개발 되었는가 | | |
| 내용 | | | |

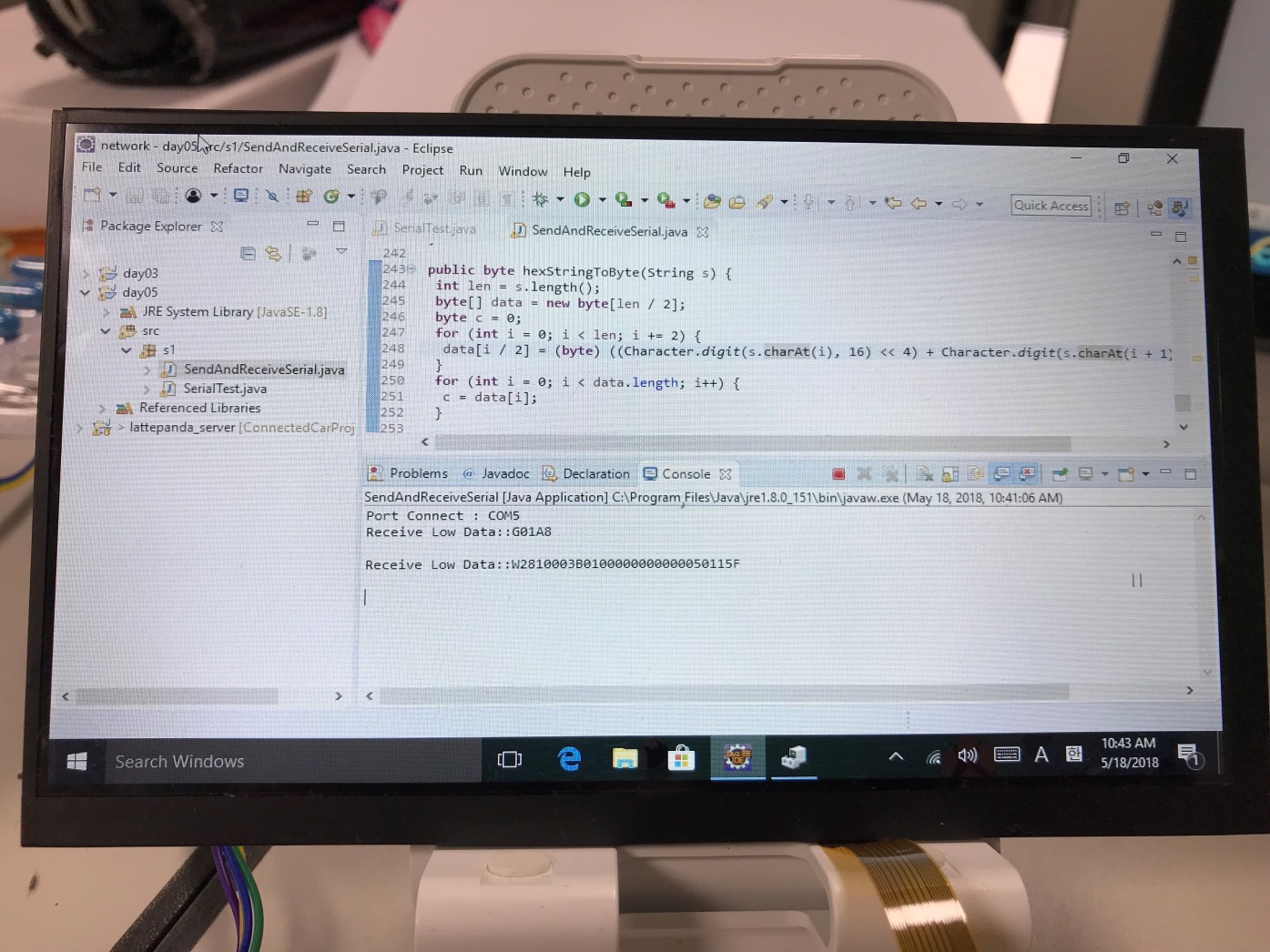
|  |
| --- |
| **시스템 구성도** |



|  |
| --- |
| **시스템 구성 사진** |







|  |
| --- |
| **프로그램 소스** |
|  |
| package s1;  import java.io.BufferedInputStream;  import java.io.IOException;  import java.io.InputStream;  import java.io.OutputStream;  import gnu.io.CommPort;  import gnu.io.CommPortIdentifier;  import gnu.io.SerialPort;  import gnu.io.SerialPortEvent;  import gnu.io.SerialPortEventListener;  public class SendAndReceiveSerial implements SerialPortEventListener {  private BufferedInputStream bin;  private InputStream in;  private OutputStream out;  private SerialPort serialPort;  private CommPortIdentifier portIdentifier;  private CommPort commPort;  private String result;  private String rawCanID, rawTotal;  //private boolean start = false;  public SendAndReceiveSerial(String portName, boolean mode) {  try {  if(mode == true){  portIdentifier = CommPortIdentifier.getPortIdentifier(portName);  System.out.printf("Port Connect : %s\n", portName);  connectSerial();  // Serial Initialization ....  (new Thread(new SerialWriter())).start();  }    } catch (Exception e) {  e.printStackTrace();  }  }  public void connectSerial() throws Exception {  if (portIdentifier.isCurrentlyOwned()) {  System.out.println("Error: Port is currently in use");  } else {  commPort = portIdentifier.open(this.getClass().getName(), 5000);  if (commPort instanceof SerialPort) {  serialPort = (SerialPort) commPort;  serialPort.addEventListener(this);  serialPort.notifyOnDataAvailable(true);  serialPort.setSerialPortParams(921600, // 통신속도  SerialPort.DATABITS\_8, // 데이터 비트  SerialPort.STOPBITS\_1, // stop 비트  SerialPort.PARITY\_NONE); // 패리티  in = serialPort.getInputStream();  bin = new BufferedInputStream(in);  out = serialPort.getOutputStream();  } else {  System.out.println("Error: Only serial ports are handled by this example.");  }  }  }  public void sendSerial(String rawTotal, String rawCanID) {  this.rawTotal = rawTotal;  this.rawCanID = rawCanID;  // System.out.println("send: " + rawTotal);  try {  // Thread.sleep(50);  Thread.sleep(30);  } catch (InterruptedException e) {  e.printStackTrace();  }  Thread sendTread = new Thread(new SerialWriter(rawTotal));  sendTread.start();  }  private class SerialWriter implements Runnable {  String data;  public SerialWriter() {  this.data = ":G11A9\r";  }  public SerialWriter(String serialData) {  //checkSum Data생성  this.data = sendDataFormat(serialData);  }      //데이터를 can데이터 형식에 맞게 바꿔준다.  //맨 뒤에 checksSum 붙여준다.  public String sendDataFormat(String serialData) {  serialData = serialData.toUpperCase();  char c[] = serialData.toCharArray();  int cdata = 0;  for (char cc : c) {  cdata += cc;  }  cdata = (cdata & 0xFF);  String returnData = ":";  returnData += serialData + Integer.toHexString(cdata).toUpperCase();  returnData += "\r";  return returnData;  }  public void run() {  try {  byte[] inputData = data.getBytes();  out.write(inputData);  } catch (IOException e) {  e.printStackTrace();  }  }  }  public byte[] getResult() {  try {  Thread.sleep(80);  } catch (InterruptedException e) {  e.printStackTrace();  }  return getResult(this.result);  }  public String webData(){  try {  Thread.sleep(80);  } catch (InterruptedException e) {  e.printStackTrace();  }  return result;  }    // Asynchronized Receive Data  // --------------------------------------------------------  // 이벤트가 발생하면 호출되는 함수  // serial port Event를 전하는 함수.  // 인터페이스 SerialPortEventListener를 확장하기 때문에 반드시 구현해야 하는 함수.  public void serialEvent(SerialPortEvent event) {  switch (event.getEventType()) {  case SerialPortEvent.BI:  case SerialPortEvent.OE:  case SerialPortEvent.FE:  case SerialPortEvent.PE:  case SerialPortEvent.CD:  case SerialPortEvent.CTS:  case SerialPortEvent.DSR:  case SerialPortEvent.RI:  case SerialPortEvent.OUTPUT\_BUFFER\_EMPTY:  break;  case SerialPortEvent.DATA\_AVAILABLE:  byte[] readBuffer = new byte[128];  try {  while (bin.available() > 0) {  int numBytes = bin.read(readBuffer);  }  String ss = new String(readBuffer);  System.out.println("Receive Low Data:" + ss + "||");    } catch (Exception e) {  e.printStackTrace();  }  break;  }  }  public boolean checkSerialData(String data) {  boolean check = false;  // :U2800000050000000000000002046  String checkData = data.substring(1, 28);  String checkSum = data.substring(28, 30);  char c[] = checkData.toCharArray();  int cdata = 0;  for (char cc : c) {  cdata += cc;  }  cdata = (cdata & 0xFF);  String serialCheckSum = Integer.toHexString(cdata).toUpperCase();  if (serialCheckSum.trim().equals(checkSum)) {  check = true;  }  return check;  }  public byte[] getResult(String result) {  byte hexData[] = null;  // ex : :U08 00000000 0102030405060708 A3    String serialData = result.substring(4, 28);  serialData = serialData.substring(0, 2) + " " + serialData.substring(2, 4) + " " + serialData.substring(4, 6)  + " " + serialData.substring(6, 8) + " " + serialData.substring(8, 10) + " "  + serialData.substring(10, 12) + " " + serialData.substring(12, 14) + " " + serialData.substring(14, 16)  + " " + serialData.substring(16, 18) + " " + serialData.substring(18, 20) + " "  + serialData.substring(20, 22) + " " + serialData.substring(22, 24);  hexData = getData(serialData, 12);  return hexData;  }  public void close() throws IOException {  try {  Thread.sleep(100);  } catch (InterruptedException e) {  e.printStackTrace();  }  if(in != null){  in.close();  }  if(out != null){  out.close();  }  if(commPort != null){  commPort.close();  }    }  // Convert Can Data  // ---------------------------------------------------------  public byte[] getData(String str, int size) {  byte result[] = new byte[size];  String[] st = str.split(" ");  for (int i = 0; i < st.length; i++) {  result[i] = hexStringToByte(st[i]);  }  return result;  }  public byte hexStringToByte(String s) {  int len = s.length();  byte[] data = new byte[len / 2];  byte c = 0;  for (int i = 0; i < len; i += 2) {  data[i / 2] = (byte) ((Character.digit(s.charAt(i), 16) << 4) + Character.digit(s.charAt(i + 1), 16));  }  for (int i = 0; i < data.length; i++) {  c = data[i];  }  return c;  }  public static void main(String args[]) throws IOException {  SendAndReceiveSerial ss = new SendAndReceiveSerial("COM7",true);    ss.sendSerial("W2810003B010000000000005011", "10003B01");  }  } | |