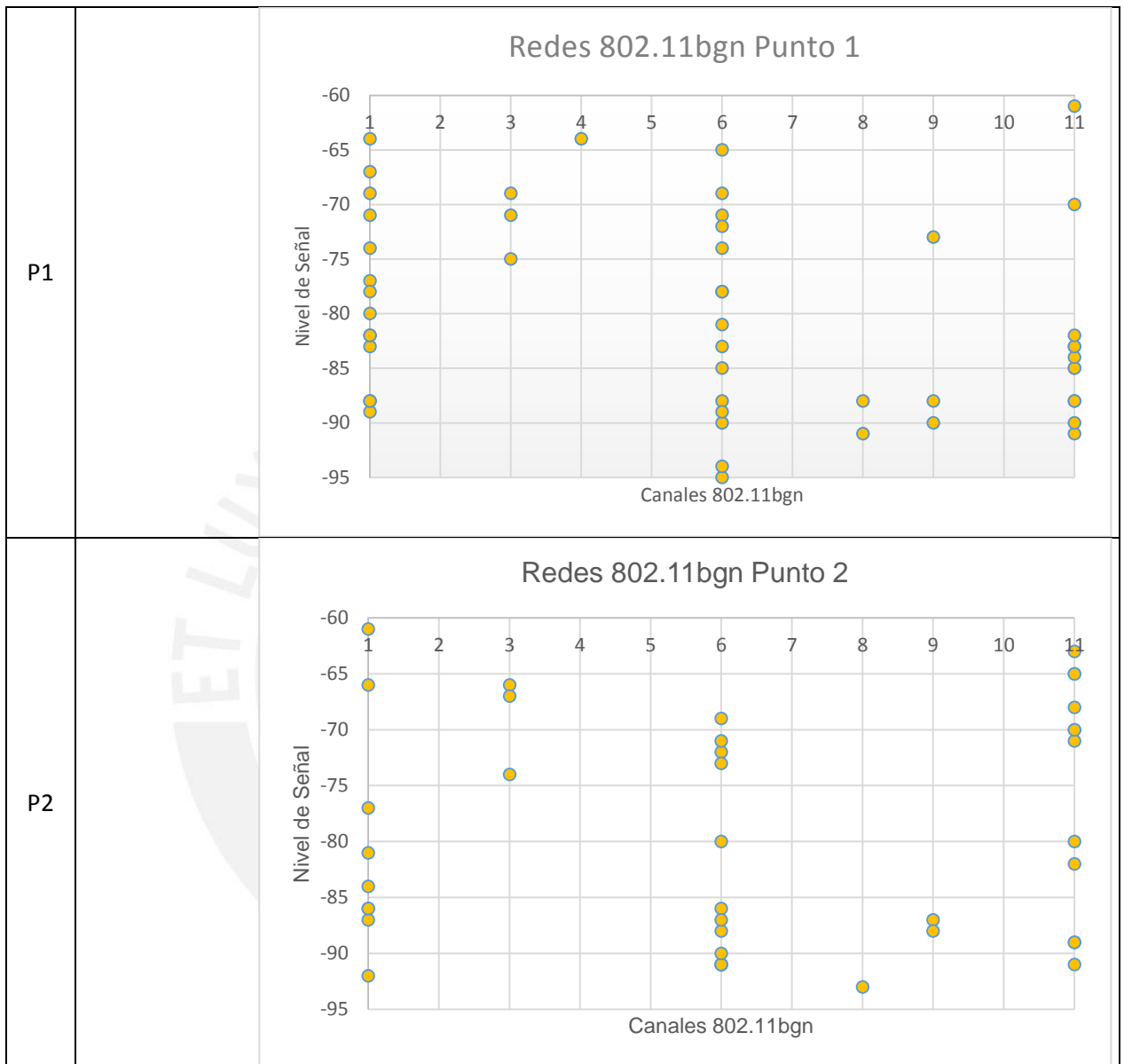
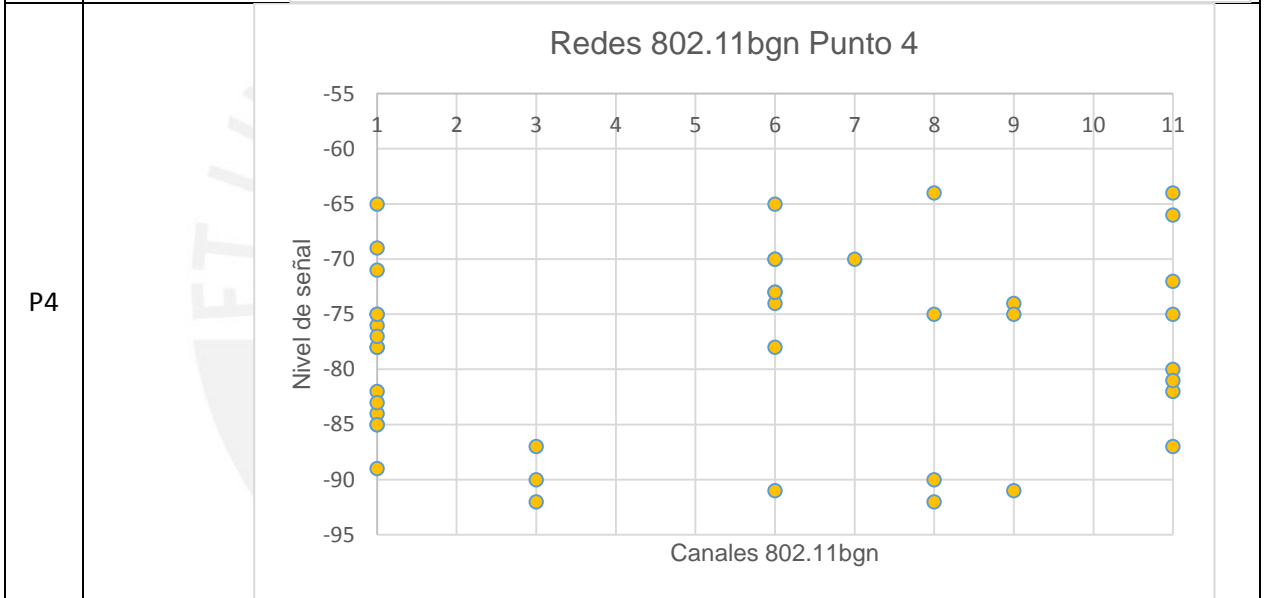
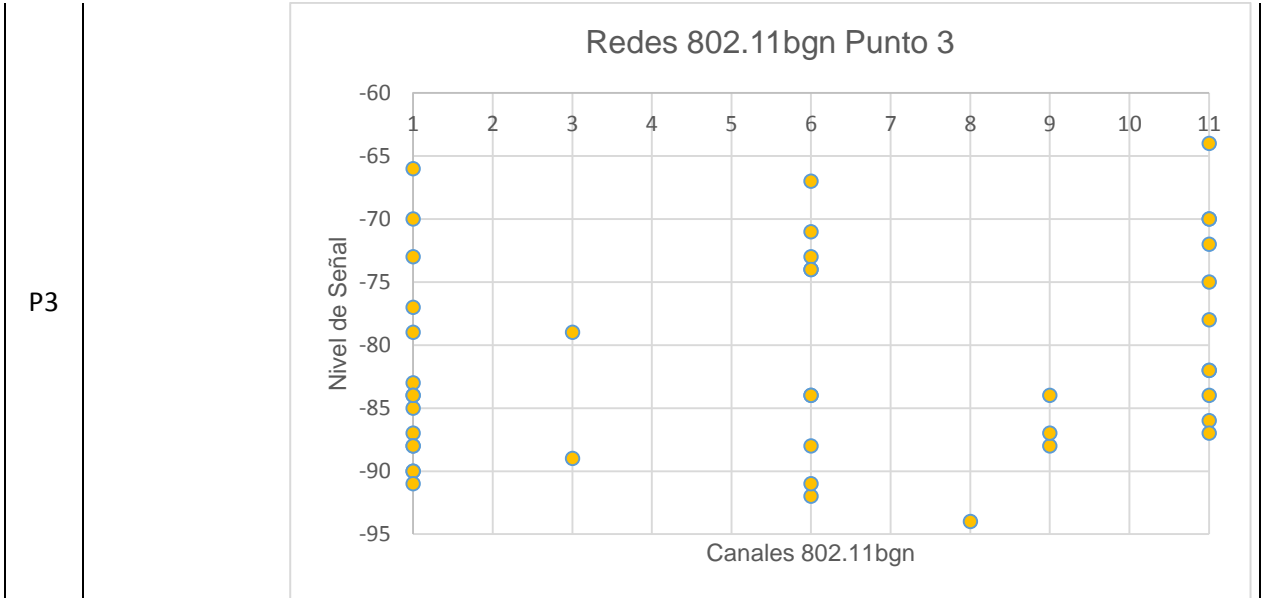


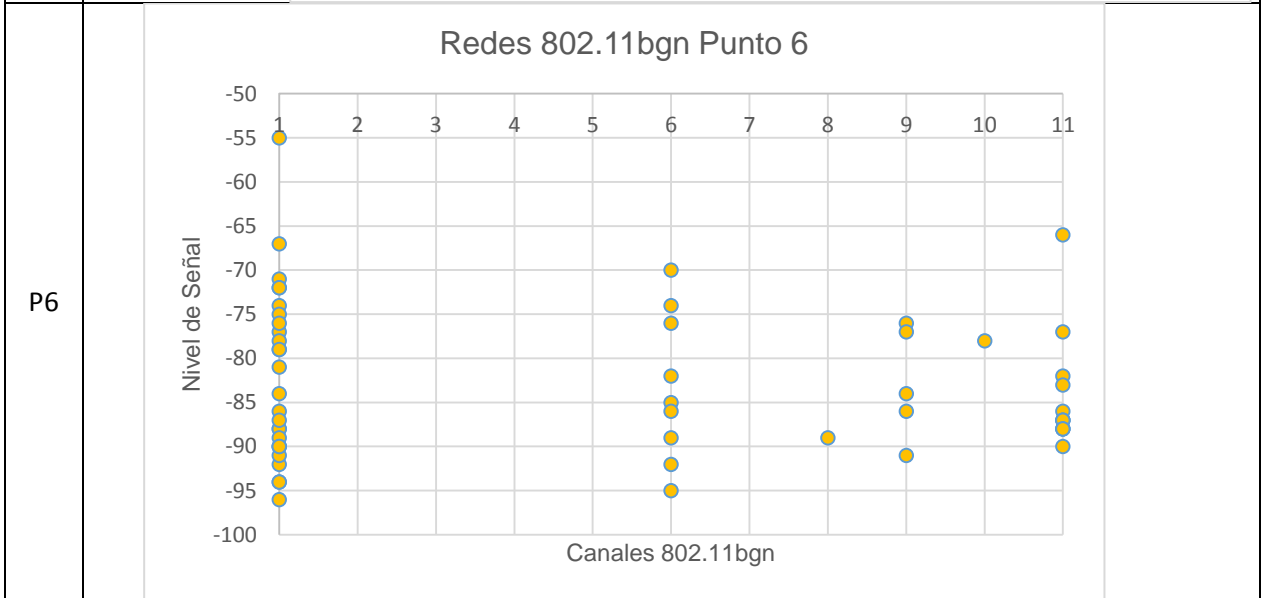
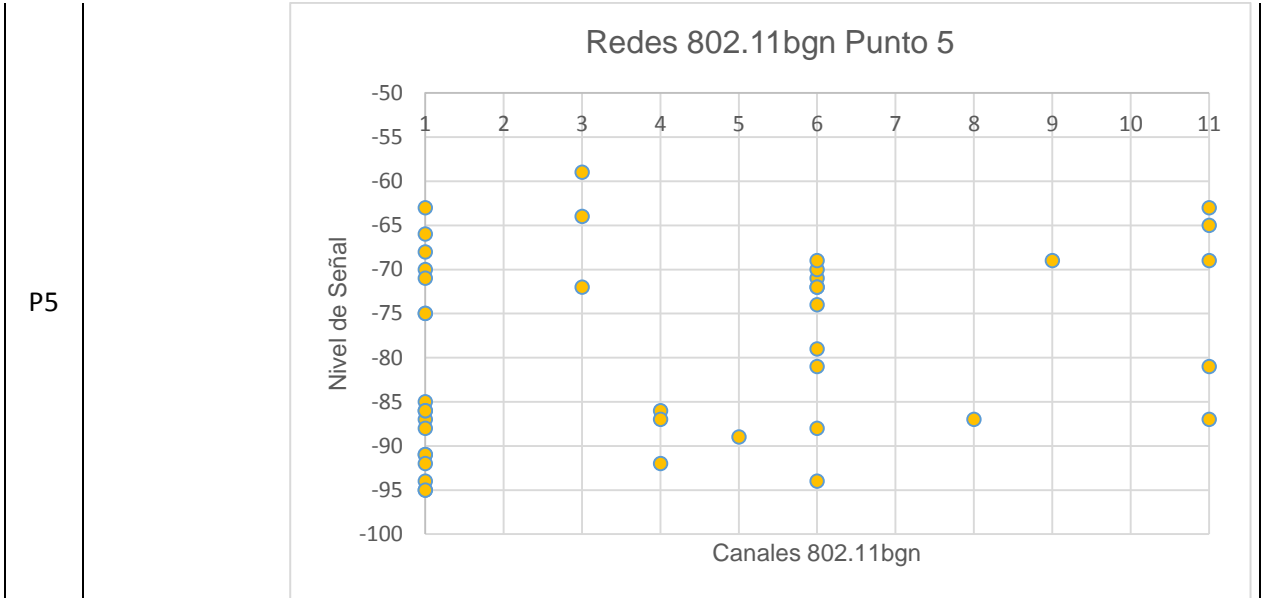
Anexos



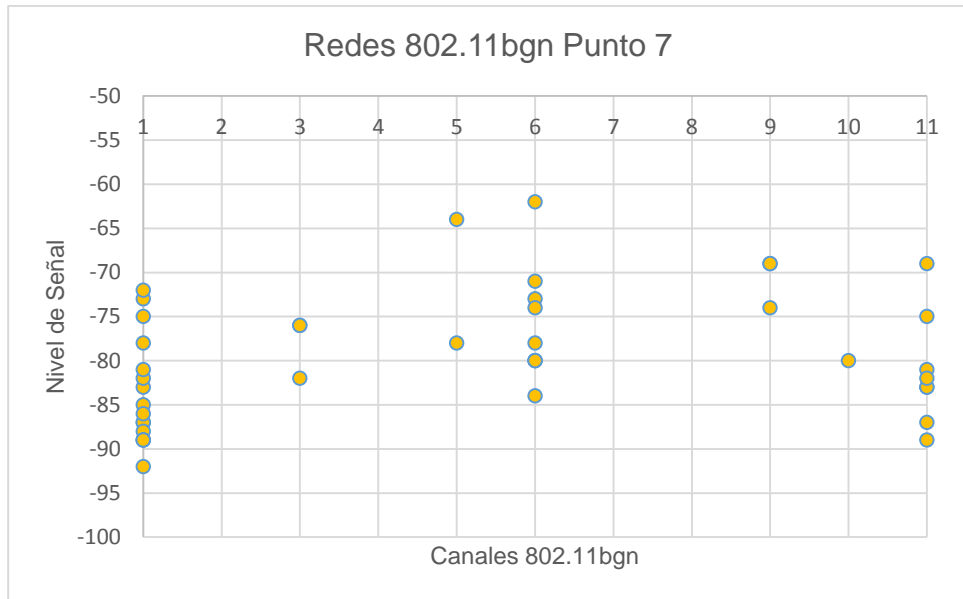
1. Intensidad de señal de los AP en la Av. La Mar por cada punto de medición.



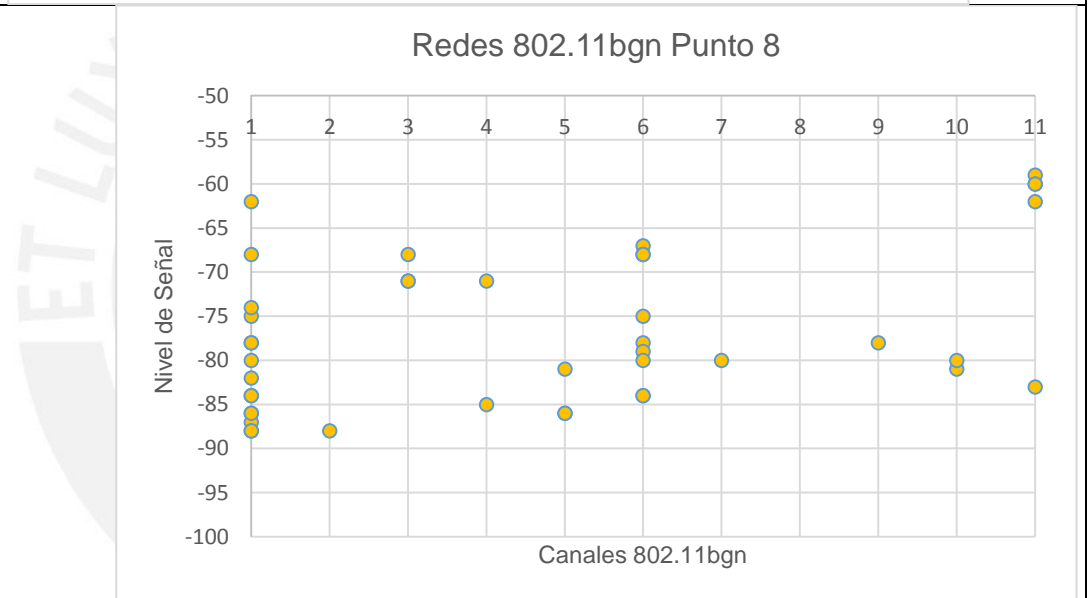




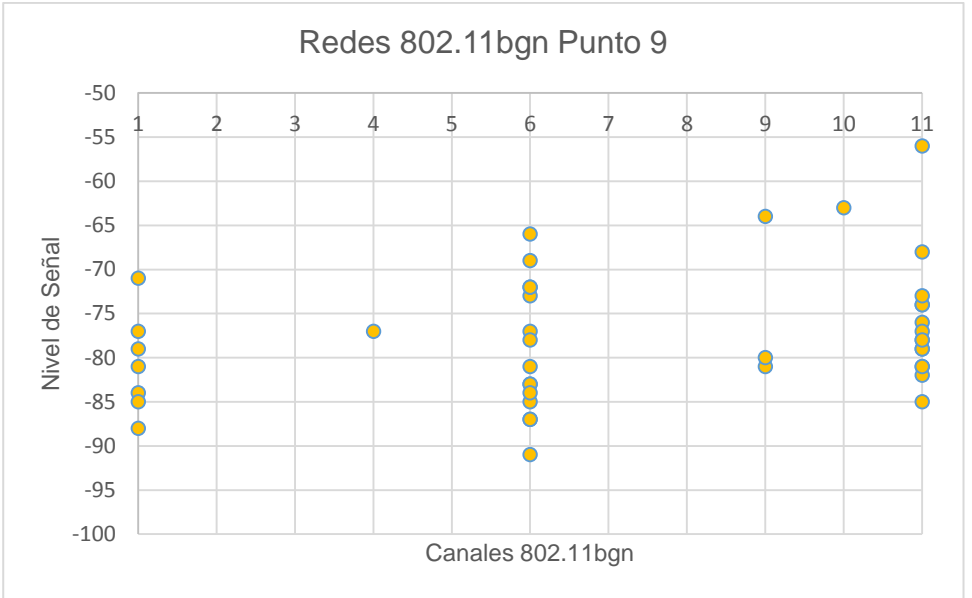
P7



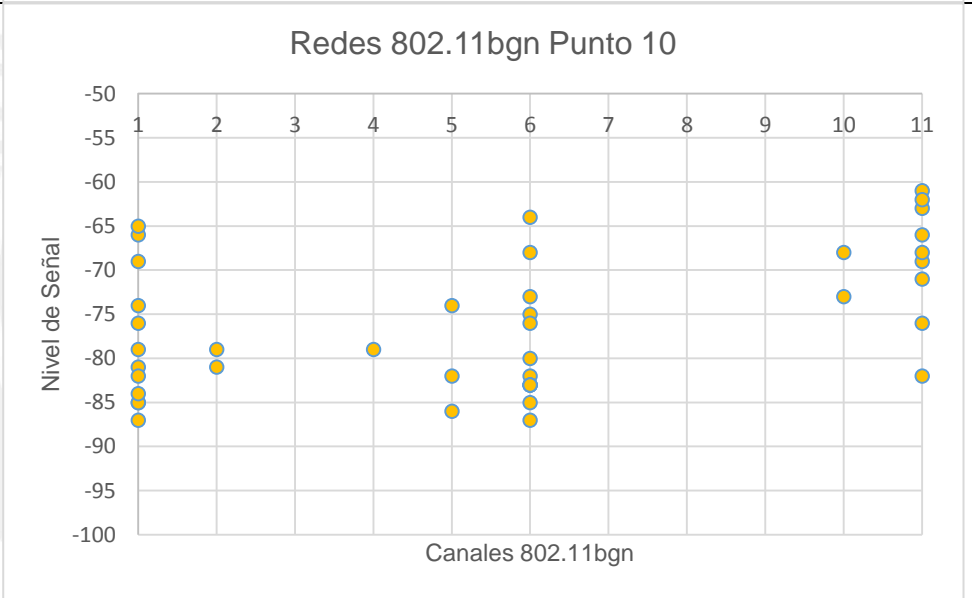
P8

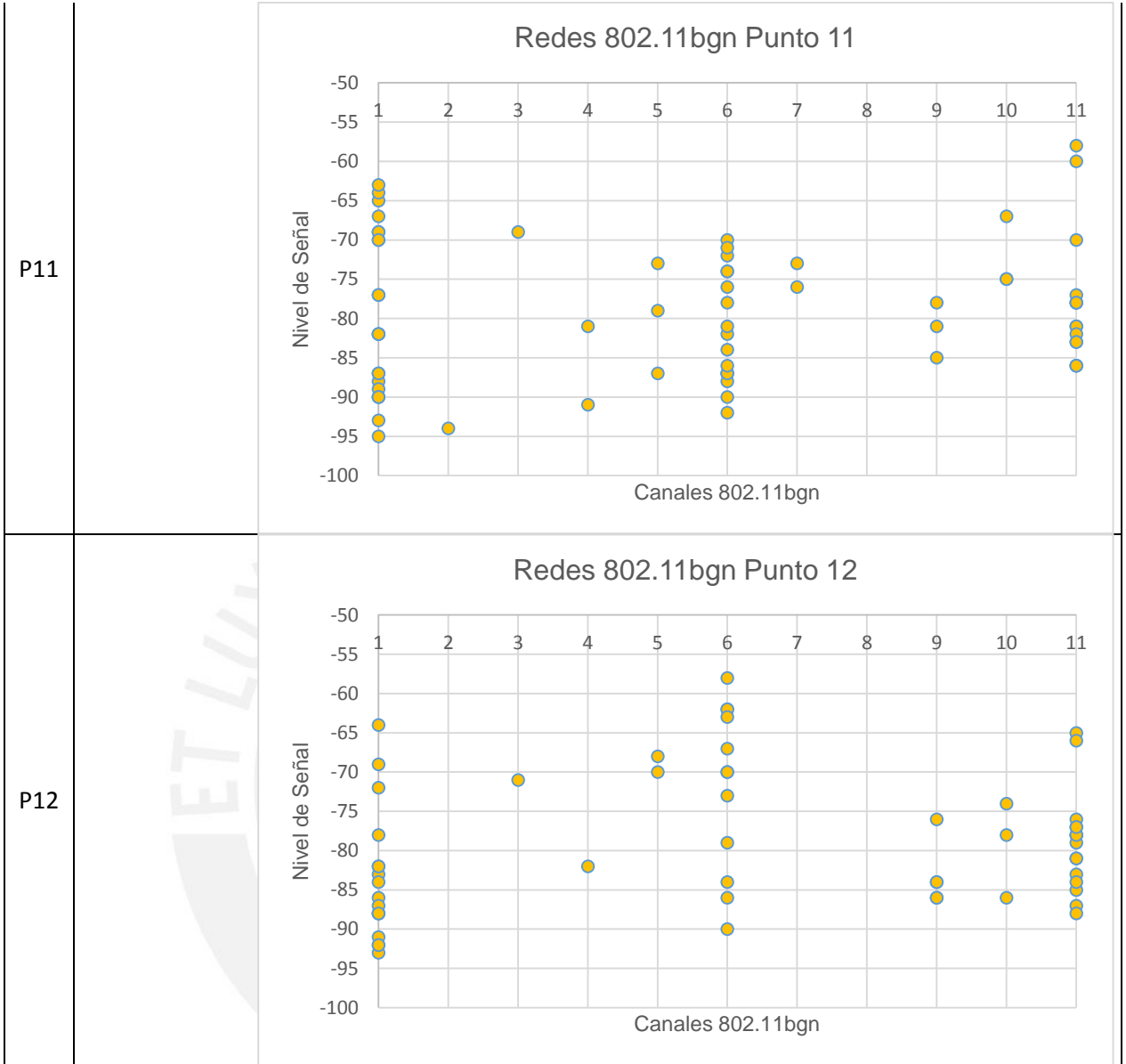


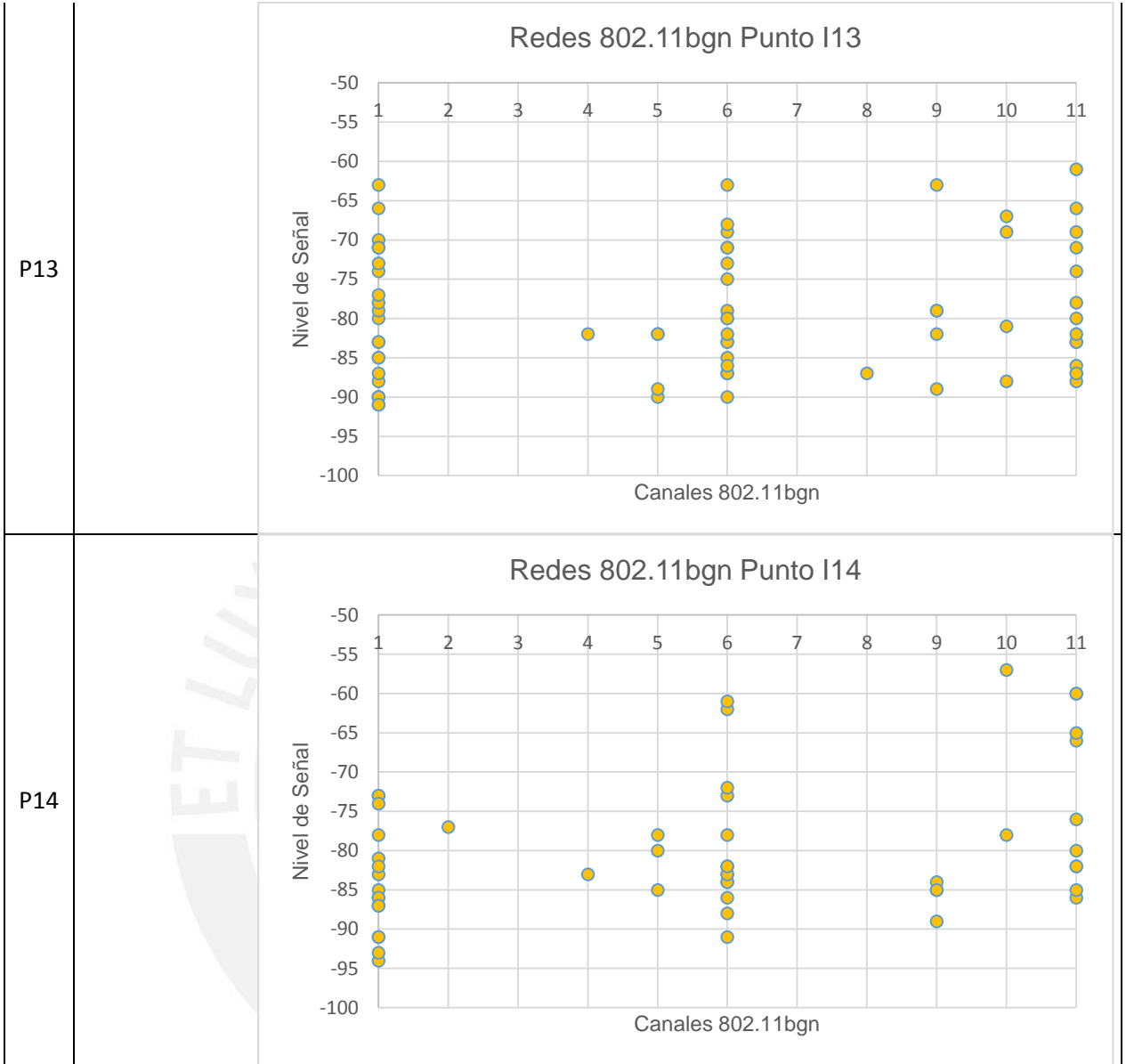
P9

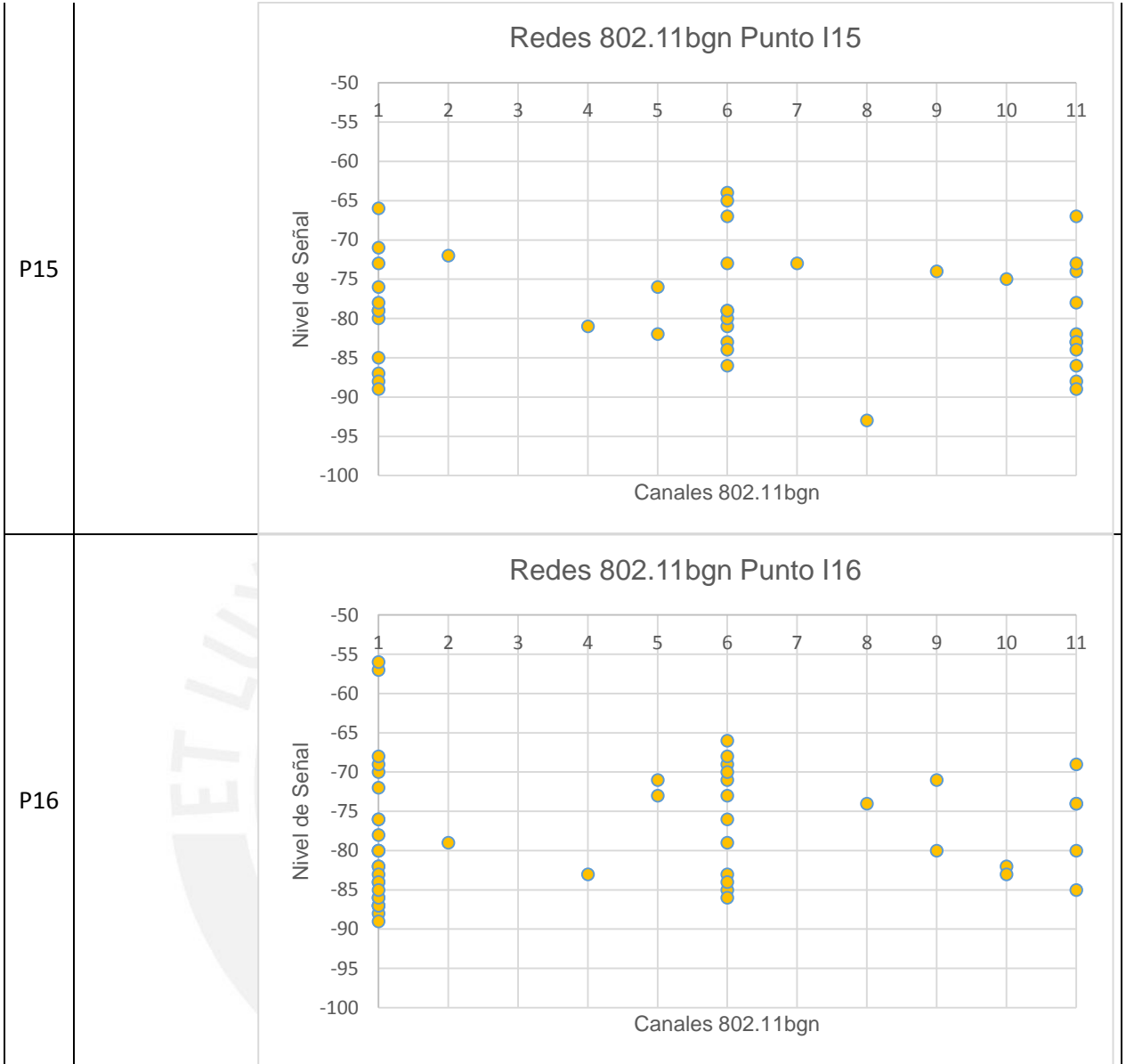


P10

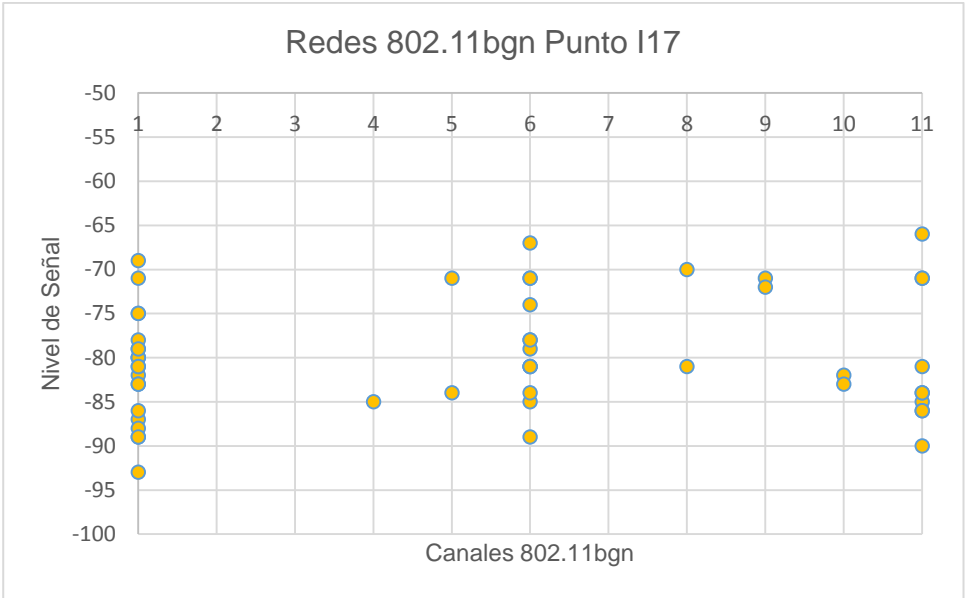




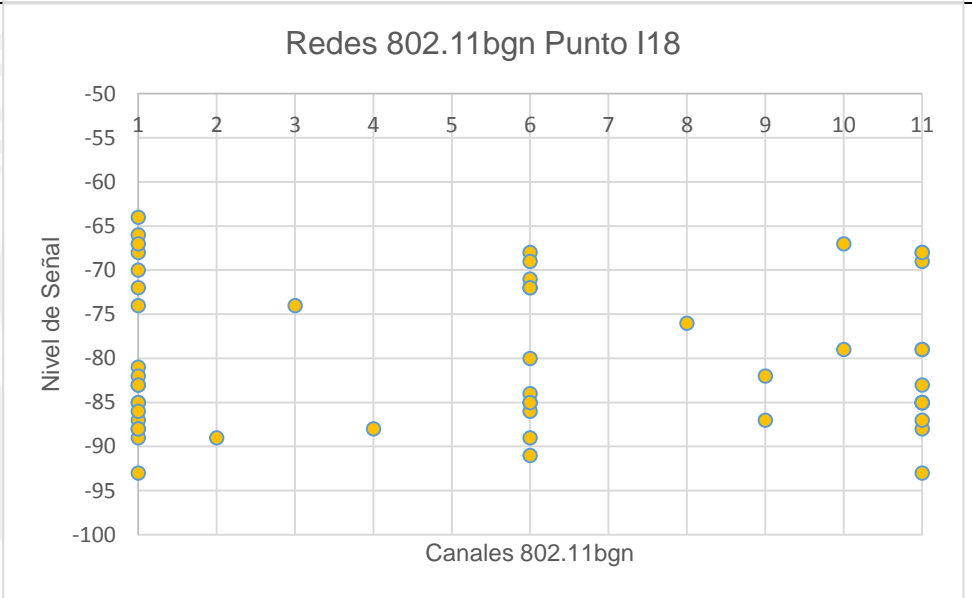


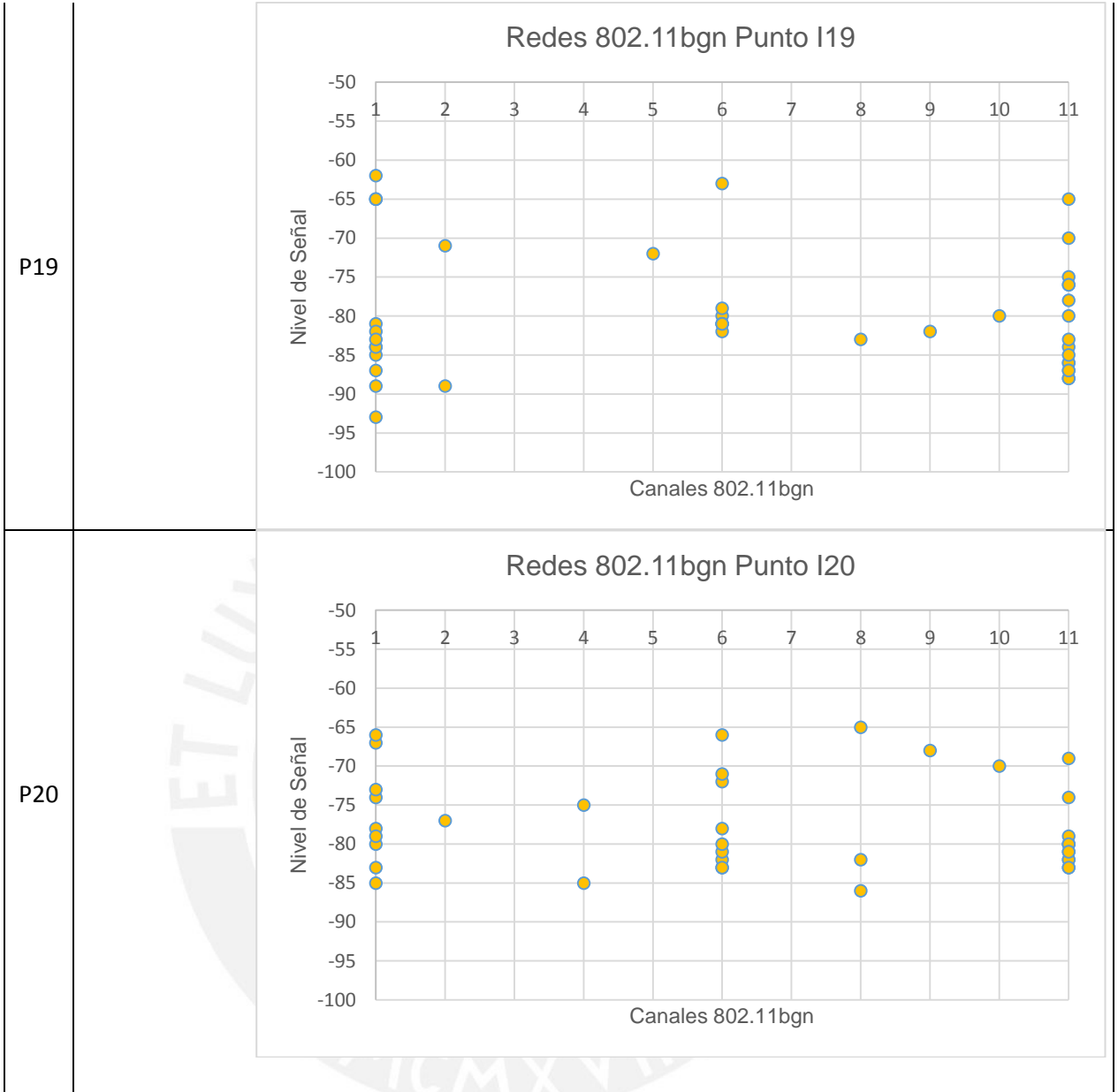


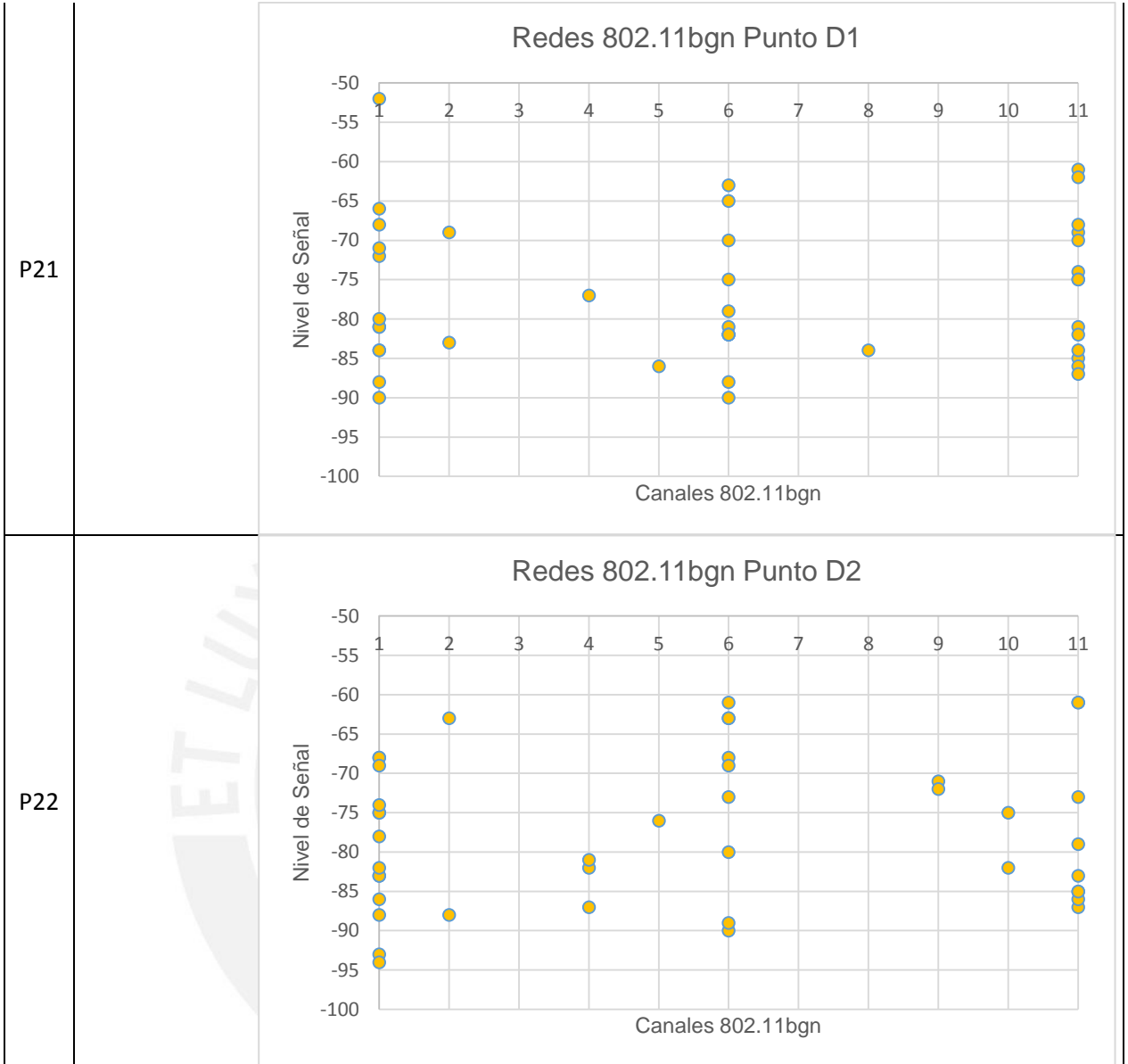
P17

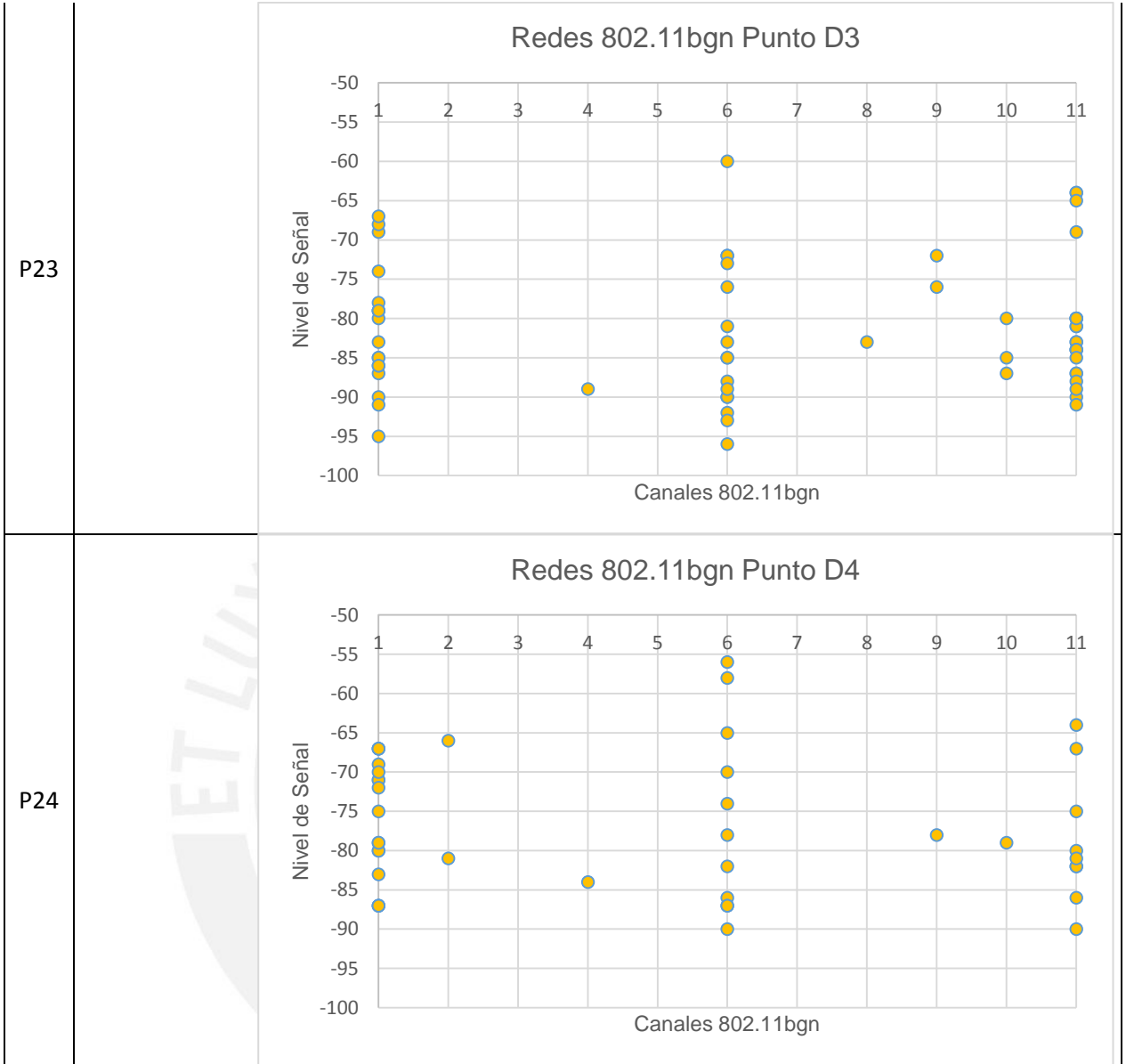


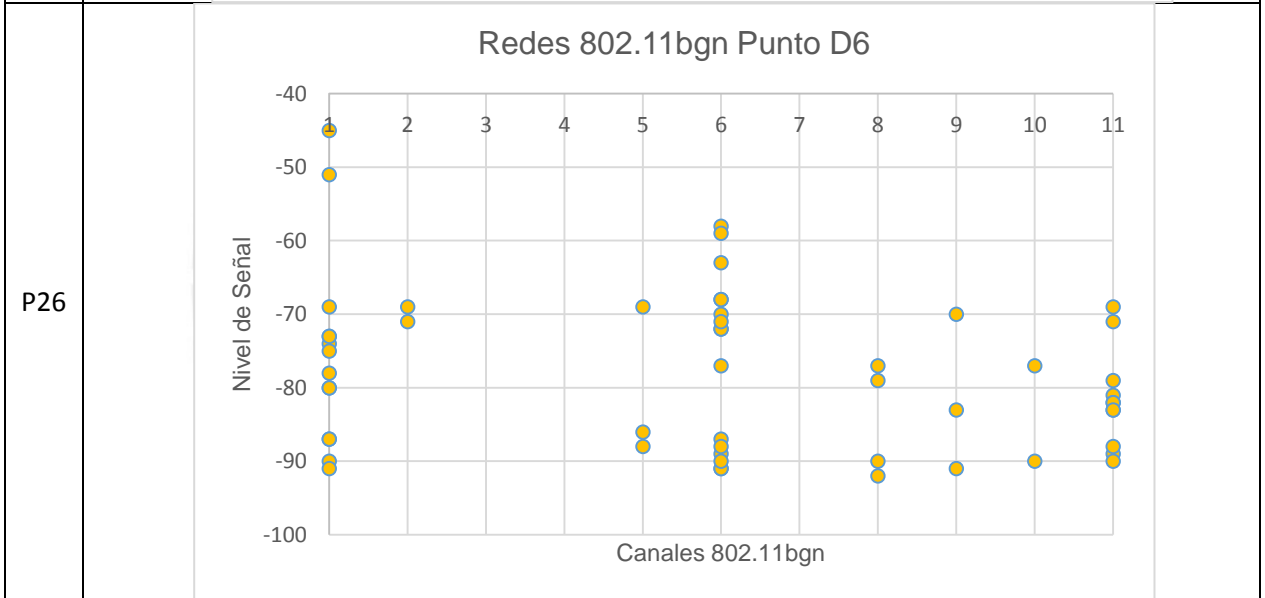
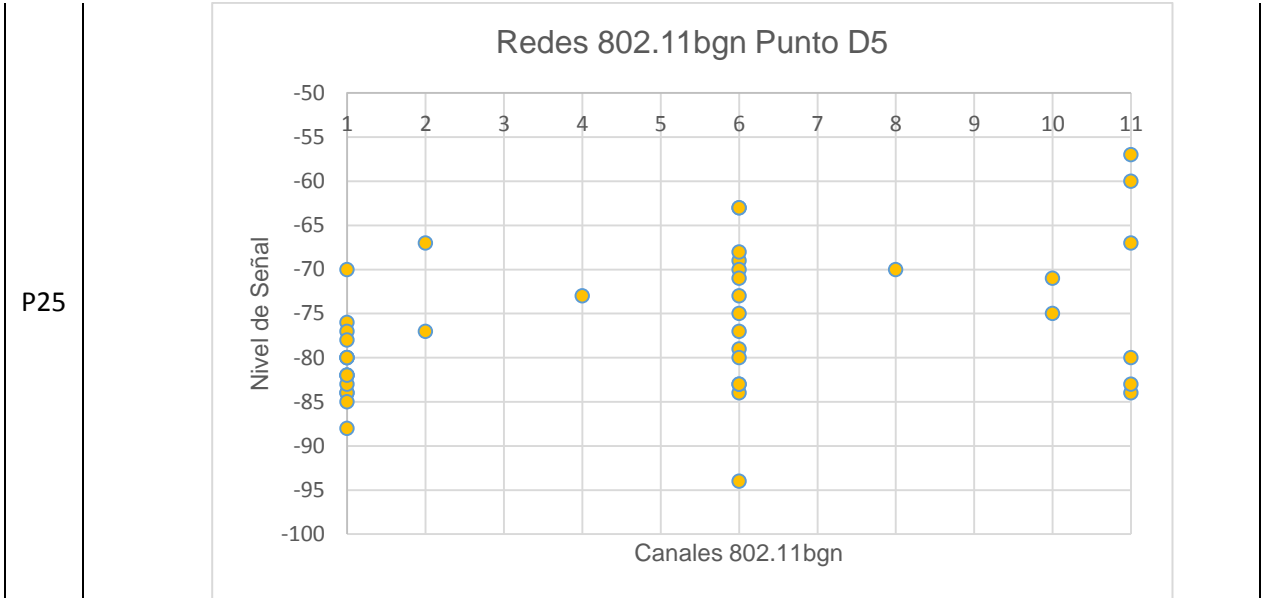
P18



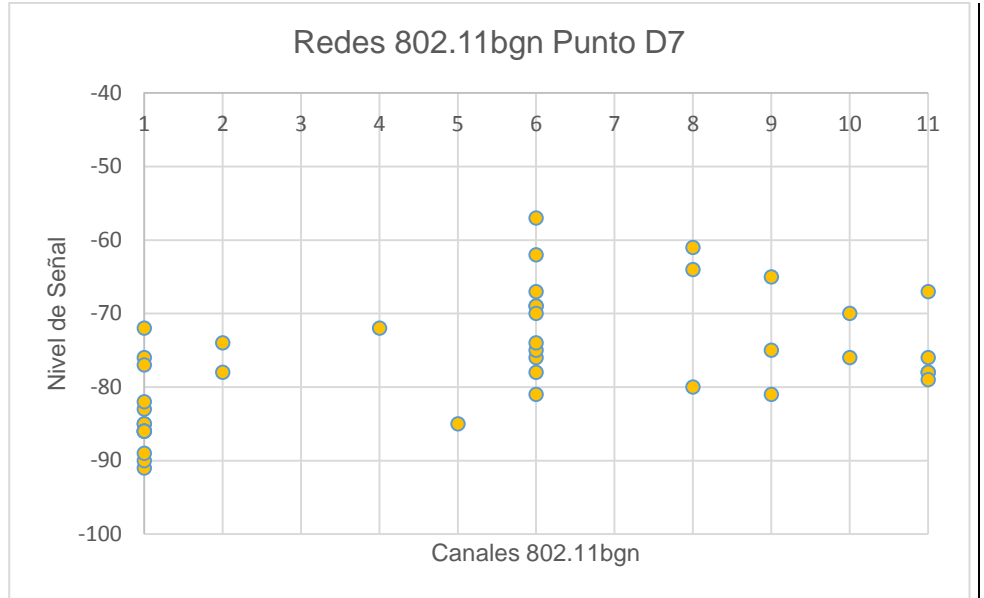




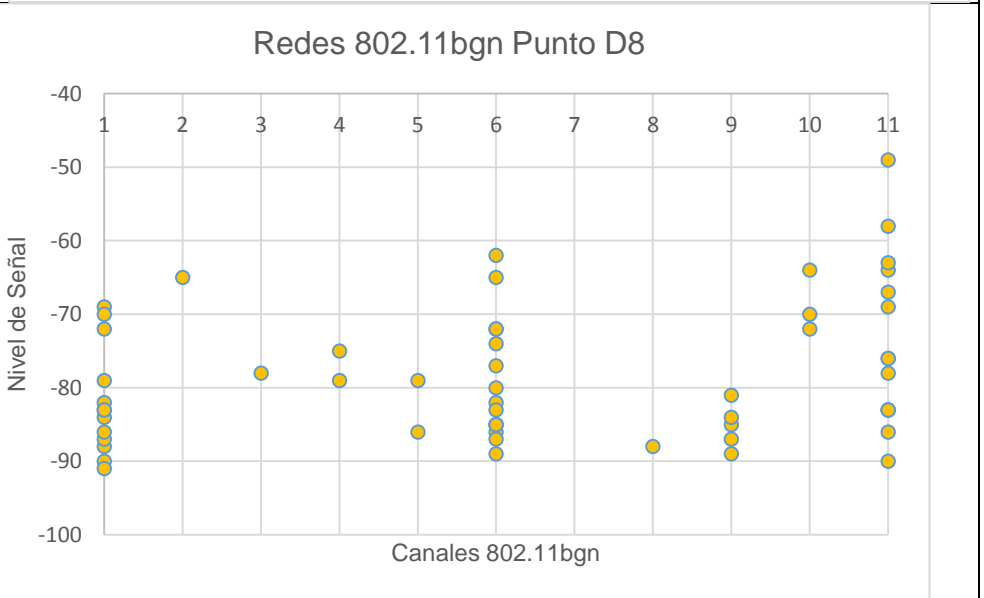


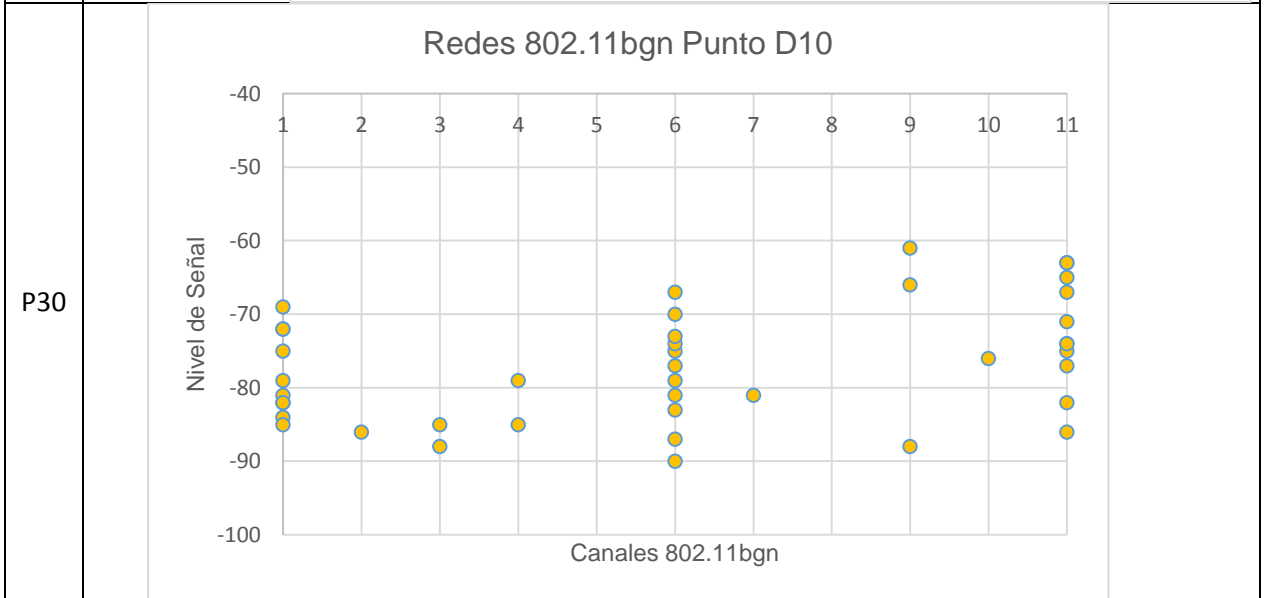
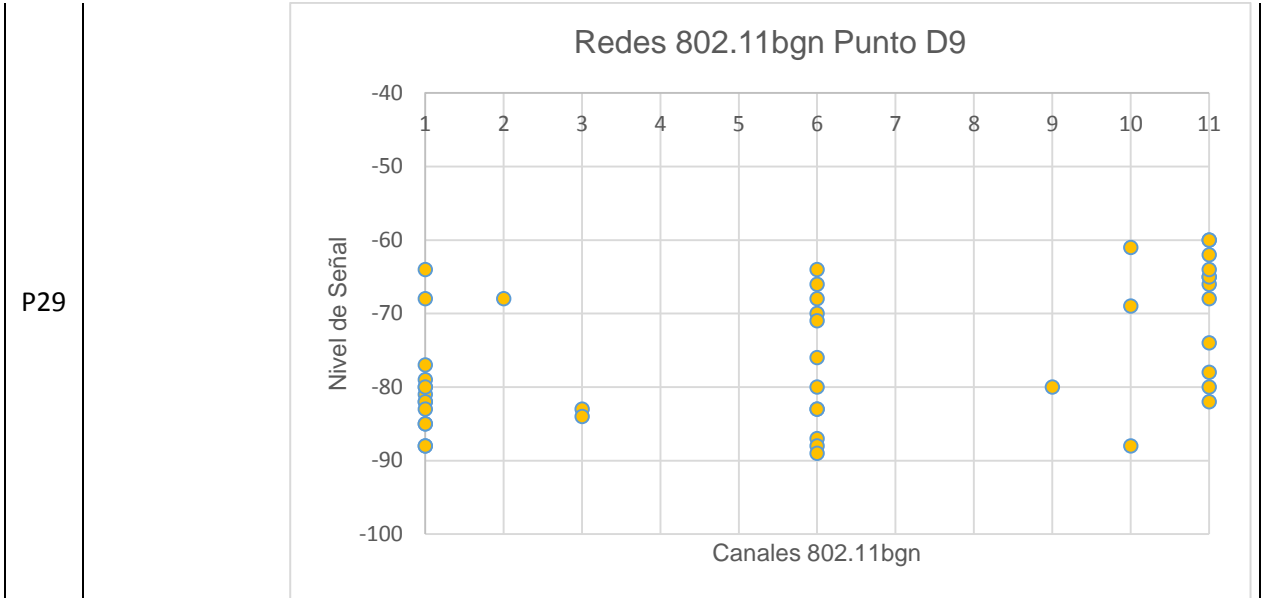


P27

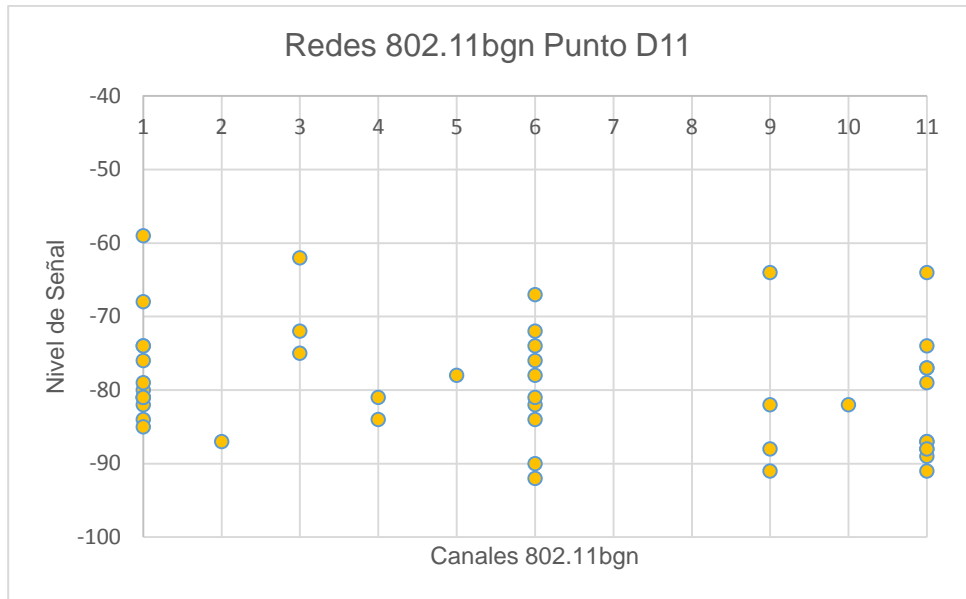


P28

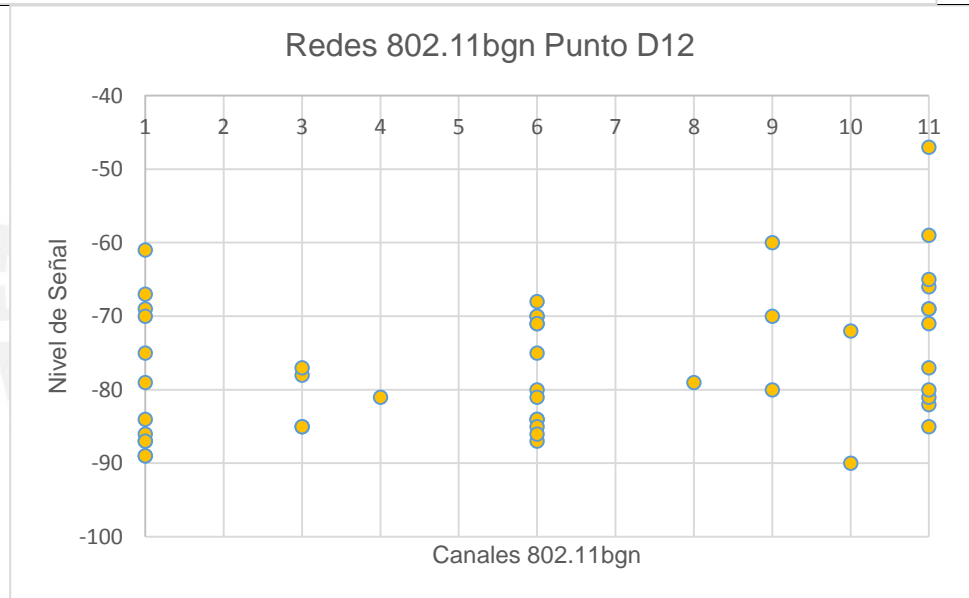


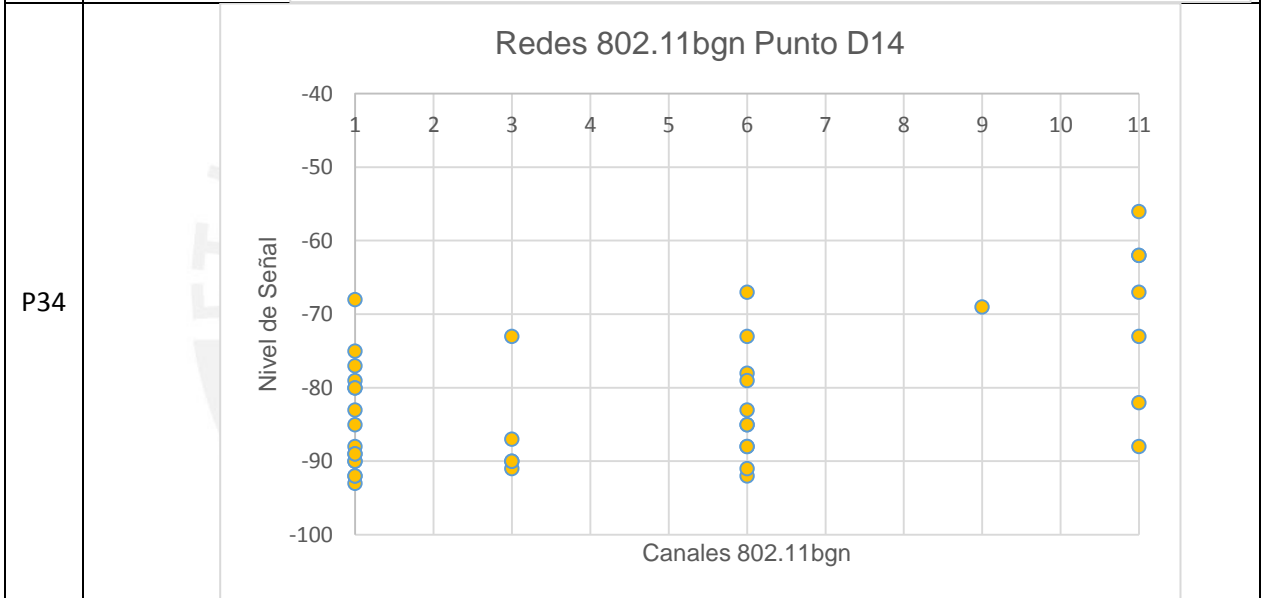
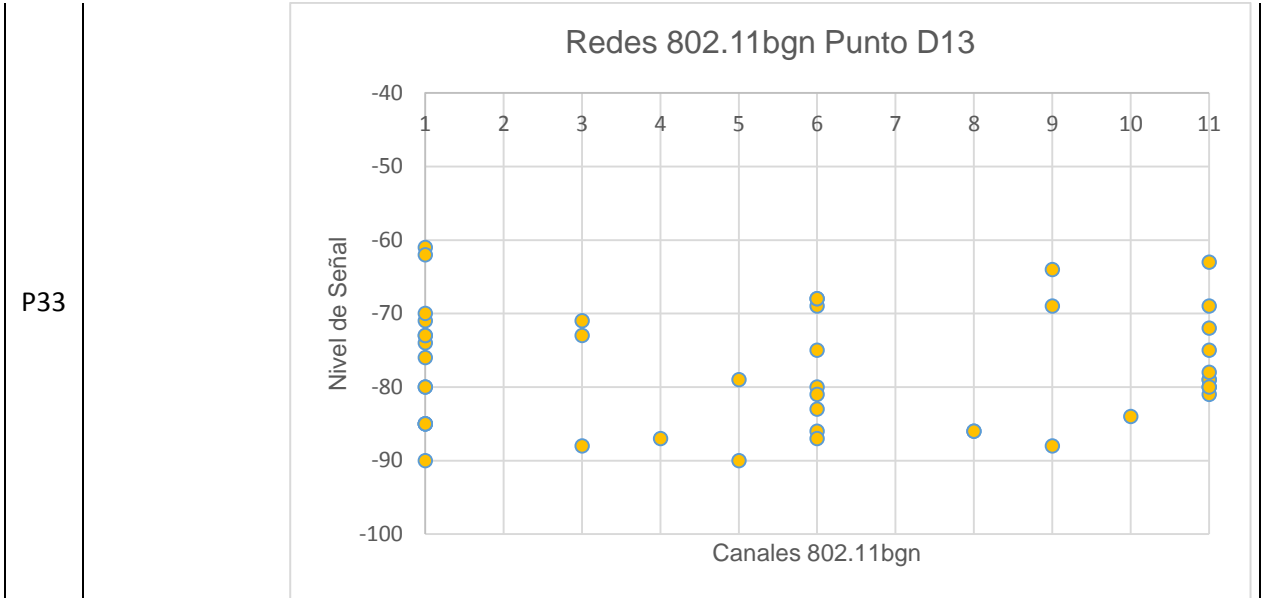


P31

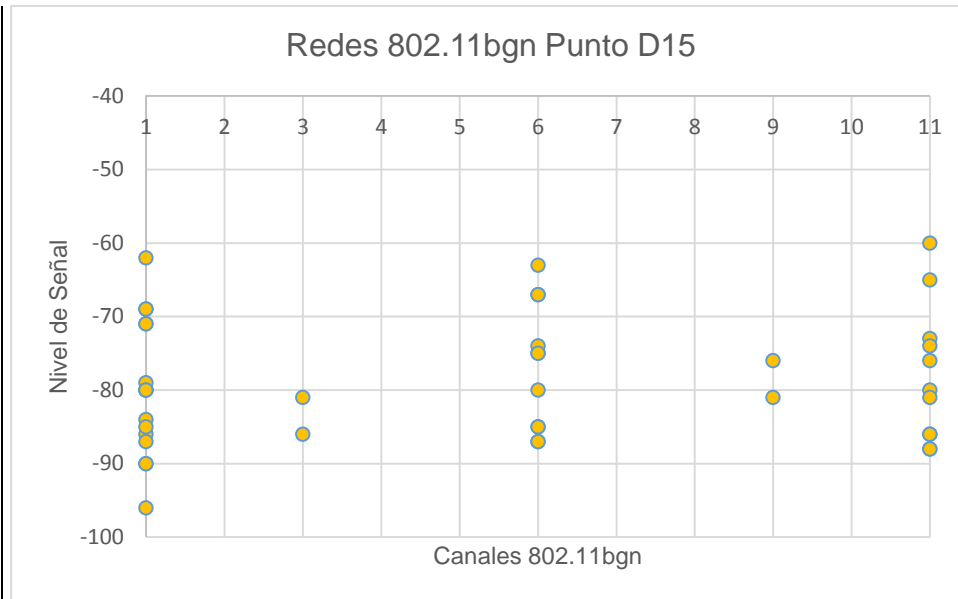


P32





P35



2. Código del programa en Java utilizado para enviar tramas entre dos radios XBee y calcular tiempos.

```

2. import javax.comm.*;
3. import java.io.*;
4. import javax.comm.*;
5. import java.io.*;
6. import java.util.concurrent.TimeoutException;
7.
8. public class Rs232 implements Runnable, SerialPortEventListener {
9.
10.     private static OutputStream os;
11.     private InputStream is;
12.     private boolean open;
13.     private CommPortIdentifier portId;
14.     private SerialPort sPort;
15.     private SerialParameters parameters;
16.
17.
18.     public Rs232() {
19.         parameters = new SerialParameters("COM11", 9600, 0, 0, 8, 1, 0);
20.         try {
21.             openConnection();
22.         } catch (SerialConnectionException e) {
23.             System.out.println("Error opening port!" + e.getMessage() + ". Select new settings, try again.");
24.             e.printStackTrace();
25.         } catch (Exception e){
26.             e.printStackTrace();
27.         }
28.
29.
30.         Thread thread = new Thread(this);

```

```

31.     thread.start();
32.
33. }
34.
35. public void run() {
36.     try {
37.         while (true/*status == DOWNLOADING*/) {
38.             Thread.sleep(5000);
39.         }
40.     } catch (Exception e) {
41.         e.printStackTrace();
42.     } finally {
43.     }
44. }
45.
46. public void openConnection() throws SerialConnectionException {
47.
48.     try {
49.         portId = CommPortIdentifier.getPortIdentifier(parameters.getPortName());
50.
51.     } catch (NoSuchPortException e) {
52.         throw new SerialConnectionException(e.getMessage());
53.     } catch (Exception e) {
54.         System.out.println(e.getMessage());
55.     }
56.
57.     try {
58.         sPort = (SerialPort) portId.open("SerialDemo", 30000);
59.         System.out.println("Conexion a puerto Serial OK!");
60.     } catch (PortInUseException e) {
61.         throw new SerialConnectionException(e.getMessage());
62.     }
63.
64.     try {
65.         setConnectionParameters();
66.     } catch (SerialConnectionException e) {
67.         sPort.close();
68.         throw e;
69.     }
70.
71.     try {
72.         os = sPort.getOutputStream();
73.         is = sPort.getInputStream();
74.     } catch (IOException e) {
75.         sPort.close();
76.         throw new SerialConnectionException("Error opening i/o streams");
77.     }
78.
79.     try {
80.         sPort.addEventListener(this);
81.     } catch (TooManyListenersException e) {
82.         sPort.close();
83.         throw new SerialConnectionException("too many listeners added");
84.     }
85.
86.     sPort.notifyOnDataAvailable(true);
87.
88.     sPort.notifyOnBreakInterrupt(true);
89.
90.     try {
91.         sPort.enableReceiveTimeout(30);

```

```

92.     } catch (UnsupportedCommOperationException e) {
93.     }
94.
95.     open = true;
96. }
97.
98. public void setConnectionParameters() throws SerialConnectionException {
99.
100.    int oldBaudRate = sPort.getBaudRate();
101.    int oldDatabits = sPort.getDataBits();
102.    int oldStopbits = sPort.getStopBits();
103.    int oldParity = sPort.getParity();
104.    int oldFlowControl = sPort.getFlowControlMode();
105.
106.    try {
107.        sPort.setSerialPortParams(parameters.getBaudRate(),
108.            parameters.getDatabits(),
109.            parameters.getStopbits(),
110.            parameters.getParity());
111.    } catch (UnsupportedCommOperationException e) {
112.        parameters.setBaudRate(oldBaudRate);
113.        parameters.setDatabits(oldDatabits);
114.        parameters.setStopbits(oldStopbits);
115.        parameters.setParity(oldParity);
116.        throw new SerialConnectionException("Unsupported parameter");
117.    }
118.
119.    try {
120.        sPort.setFlowControlMode(parameters.getFlowControlIn()
121.            | parameters.getFlowControlOut());
122.    } catch (UnsupportedCommOperationException e) {
123.        throw new SerialConnectionException("Unsupported flow control");
124.    }
125. }
126.
127. public void closeConnection() {
128.     if (!open) {
129.         return;
130.     }
131.
132.     if (sPort != null) {
133.         try {
134.             os.close();
135.             is.close();
136.         } catch (IOException e) {
137.             System.err.println(e);
138.         }
139.         sPort.close();
140.     }
141.     open = false;
142. }
143.
144. public static boolean sendBytes(byte msg[]) {
145.     try {
146.         os.write(msg);
147.         return true;
148.     } catch (IOException e) {
149.         e.printStackTrace();
150.         return false;
151.     }
152. }

```

```

153.
154. public boolean isOpen() {
155.     return open;
156. }
157.
158. public void serialEvent(SerialPortEvent e) {
159.     StringBuffer inputBuffer = new StringBuffer();
160.     int newData = 0;
161.
162.     switch (e.getEventType()) {
163.
164.         case SerialPortEvent.DATA_AVAILABLE:
165.             byte[] b ;
166.             b = new byte[808];
167.             int i = 0, idMessage=0;
168.
169.             while (newData != -1) {
170.                 try {
171.                     newData = is.read();
172.                     if (newData == -1) {
173.                         break;
174.                     }
175.
176.                     if ('\r' != (char) newData) {
177.                         b[i++] = (byte) newData;
178.                     } else {
179.                         byte[] datetime = new byte[8];
180.                         for (int j=0;j<8;j++){
181.                             datetime[j] = b[j];
182.                         }
183.
184.                         int i1 = b[8]<0 ? 256+b[8] : b[8];
185.                         int i0 = b[9]<0 ? 256+b[9] : b[9];
186.
187.                         idMessage += i1<<8;
188.                         idMessage += i0;
189.
190.                         long now = System.currentTimeMillis();
191.                         long llego = byteArrayToLong(datetime);
192.                         System.out.println(idMessage+", "+llego+", "+(now - llego));
193.
194.                         i=0;
195.                         b = new byte[808];
196.                         idMessage = 0;
197.                     }
198.
199.                 } catch (IOException ex) {
200.                     System.err.println(ex);
201.                     return;
202.                 }
203.             }
204.             if (inputBuffer.length() > 0) {
205.                 String message = inputBuffer.toString();
206.             }
207.             break;
208.         case SerialPortEvent.BI:
209.     }
210. }
211.
212. public long byteArrayToLong(byte[] value) {
213.     long miliseconds=0;

```

```
214. long b7 = value[7]<0 ? 256+value[7] : value[7];
215. long b6 = value[6]<0 ? 256+value[6] : value[6];
216. long b5 = value[5]<0 ? 256+value[5] : value[5];
217. long b4 = value[4]<0 ? 256+value[4] : value[4];
218. long b3 = value[3]<0 ? 256+value[3] : value[3];
219. long b2 = value[2]<0 ? 256+value[2] : value[2];
220. long b1 = value[1]<0 ? 256+value[1] : value[1];
221. long b0 = value[0]<0 ? 256+value[0] : value[0];
222.
223. milliseconds = b0<<56;
224. milliseconds += b1<<48;
225. milliseconds += b2<<40;
226. milliseconds += b3<<32;
227. milliseconds += b4<<24;
228. milliseconds += b5<<16;
229. milliseconds += b6<<8;
230. milliseconds += b7;
231.
232. return milliseconds;
233. }
234.
235. }
```

